CoreDX ™ Data Distribution Service

The leading Small Footprint DDS Middleware

C Reference Manual

Twin Oaks Computing, Inc

Castle Rock, CO 80108

Jan 2017

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

Introduction

Introduction

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

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

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

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

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

Document Organization

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 DDS 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. X-Types DynamicType
   This includes routines and objects to support **dynamic** data types.
6. CoreDX DDS Transports
   CoreDX DDS Tranpsorts

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

Intended Audience

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

1.1 DDS Entities

Data Structures

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

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

1.1.1 Detailed Description

1.2 DDS Quality of Service

Data Structures

- 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.

Macros

- #define DDS_XCDR_DATA_REPRESENTATION 0 /* type: DDS_DataRepresentationId_t */

  a DDS_DataRepresentationId_t that indicates XCDR form.
- #define DDS_XML_DATA_REPRESENTATION 1 /* type: DDS_DataRepresentationId_t */

  a DDS_DataRepresentationId_t that indicates XML form.

Typedefs

- typedef short DDS_DataRepresentationId_t

  Indicates the form of data, for example CDR or XML.

Functions

- DDS_QosProvider * DDS_QosProvider_newQosProvider (const char *uri, const char *profile, CDX_XmlApi *xml_parser)

  Create a new QosProvider instance that loads QoS settings from a library.
- void DDS_QosProvider_return_provider (DDS_QosProvider *qp)

  Return a QosProvider back to the factory.
- DDS_DomainParticipantFactoryQos * DDS_QosProvider_get_participantfactory_qos (DDS_QosProvider *qp, const char *id)

  Access a DomainParticipantFactory QoS.
- void DDS_QosProvider_return_participantfactory_qos (DDS_QosProvider *qp, DDS_DomainParticipantFactoryQos *qos)

  Return a DomainParticipanFactoryQos back to the factory.
- DDS_DomainParticipantQos * DDS_QosProvider_get_participant_qos (DDS_QosProvider *qp, const char *id)

  Access a DomainParticipant QoS.
- void DDS_QosProvider_return_participant_qos (DDS_QosProvider *qp, DDS_DomainParticipantQos *qos)

Return a DomainParticipantQos back to the factory.

- DDS_TopicQos * DDS_QosProvider_get_topic_qos (DDS_QosProvider *qp, const char *id)

  Access a Topic QoS.
- void DDS_QosProvider_return_topic_qos (DDS_QosProvider *qp, DDS_TopicQos *qos)

  Return a TopicQos back to the factory.
- DDS_SubscriberQos * DDS_QosProvider_get_subscriber_qos (DDS_QosProvider *qp, const char *id)

  Access a Subscriber QoS.
- void DDS_QosProvider_return_subscriber_qos (DDS_QosProvider *qp, DDS_SubscriberQos *qos)

  Return a SubscriberQos back to the factory.
- DDS_PublisherQos * DDS_QosProvider_get_publisher_qos (DDS_QosProvider *qp, const char *id)

  Access a Publisher QoS.
- void DDS_QosProvider_return_publisher_qos (DDS_QosProvider *qp, DDS_PublisherQos *qos)

  Return a PublisherQos back to the factory.
- DDS_DataReaderQos * DDS_QosProvider_get_datareader_qos (DDS_QosProvider *qp, const char *id)

  Access a DataReader QoS.
- void DDS_QosProvider_return_datareader_qos (DDS_QosProvider *qp, DDS_DataReaderQos *qos)

  Return a DataReaderQos back to the factory.
- DDS_DataWriterQos * DDS_QosProvider_get_datawriter_qos (DDS_QosProvider *qp, const char *id)

  Access a DataWriter QoS.
- void DDS_QosProvider_return_datawriter_qos (DDS_QosProvider *qp, DDS_DataWriterQos *qos)

  Return a DataWriterQos back to the factory.

1.2.1 Detailed Description

1.2.2 Function Documentation

1.2.2.1 DDS_DataReaderQos* DDS_QosProvider_get_datareader_qos (DDS_QosProvider * *qp*, const char * *id*)

Access a DataReader QoS.

If 'id' is NULL, then the first instance of DataReaderQos in the library is returned. If 'id' is not NULL, then the 'name' field of the QoS policy is checked, and the first match is returned.

1.2.2.2 DDS_DataWriterQos* DDS_QosProvider_get_datawriter_qos (DDS_QosProvider * *qp*, const char * *id*)

Access a DataWriter QoS.

If 'id' is NULL, then the first instance of DataWriterQos in the library is returned. If 'id' is not NULL, then the 'name' field of the QoS policy is checked, and the first match is returned.

1.2.2.3 DDS_DomainParticipantQos* DDS_QosProvider_get_participant_qos (DDS_QosProvider * *qp*, const char * *id*)

Access a DomainParticipant QoS.

If 'id' is NULL, then the first instance of DomainParticipantQos in the library is returned. If 'id' is not NULL, then the 'name' field of the QoS policy is checked, and the first match is returned.

1.2.2.4 DDS_DomainParticipantFactoryQos* DDS_QosProvider_get_participantfactory_qos (DDS_QosProvider * *qp*, const char * *id*)

Access a DomainParticipantFactory QoS.

If 'id' is NULL, then the first instance of DomainParticipantFactoryQos in the library is returned. If 'id' is not NULL, then the 'name' field of the DomainParticpantFactoryQos is checked, and the first match is returned.

1.2.2.5 DDS_PublisherQos* DDS_QosProvider_get_publisher_qos (DDS_QosProvider * *qp*, const char * *id*)

Access a Publisher QoS.

If 'id' is NULL, then the first instance of PublisherQos in the library is returned. If 'id' is not NULL, then the 'name' field of the QoS policy is checked, and the first match is returned.

1.2.2.6 DDS_SubscriberQos* DDS_QosProvider_get_subscriber_qos (DDS_QosProvider * *qp*, const char * *id*)

Access a Subscriber QoS.

If 'id' is NULL, then the first instance of SubscriberQos in the library is returned. If 'id' is not NULL, then the 'name' field of the QoS policy is checked, and the first match is returned.

1.2.2.7 DDS_TopicQos* DDS_QosProvider_get_topic_qos (DDS_QosProvider * *qp*, const char * *id*)

Access a Topic QoS.

If 'id' is NULL, then the first instance of TopicQos in the library is returned. If 'id' is not NULL, then the 'name' field of the QoS policy is checked, and the first match is returned.

1.2.2.8 DDS_QosProvider* DDS_QosProvider_newQosProvider (const char * *uri*, const char * *profile*, CDX_XmlApi * *xml_parser*)

Create a new QosProvider instance that loads QoS settings from a library.

The QoS library source is indicated by 'uri'. Currently CoreDX DDS supports only the '`file:///`' type URI. The QosProvider selects QoS policies from the library based on the 'profile' parameter. The 'xml_parser' paramater provides the implementation of an XML parser that should be used to parse the QoS library document. CoreDX DDS provides an implementation of CDX_XmlApi based on libxml2 named 'cdx_libxml2_impl' available in the include/dds/xml_api.h header file. Alternate implemenations are also available, including on based on RapidXML.

1.2.2.9 void DDS_QosProvider_return_datareader_qos (DDS_QosProvider * *qp*, DDS_DataReaderQos * *qos*)

Return a DataReaderQos back to the factory.

This will reclaim any resources allocated by the QosProvider.

1.2.2.10 void DDS_QosProvider_return_datawriter_qos (DDS_QosProvider * *qp*, DDS_DataWriterQos * *qos*)

Return a DataWriterQos back to the factory.

This will reclaim any resources allocated by the QosProvider.

1.2.2.11 void DDS_QosProvider_return_participant_qos (DDS_QosProvider * *qp,* DDS_DomainParticipantQos * *qos*)

Return a DomainParticipantQos back to the factory.

This will reclaim any resources allocated by the QosProvider.

1.2.2.12 void DDS_QosProvider_return_participantfactory_qos (DDS_QosProvider * *qp,* DDS_DomainParticipantFactoryQos * *qos*)

Return a DomainParticipanFactoryQos back to the factory.

This will reclaim any resources allocated by the QosProvider.

1.2.2.13 void DDS_QosProvider_return_provider (DDS_QosProvider * *qp*)

Return a QosProvider back to the factory.

This will reclaim any resources allocated by the QosProvider.

1.2.2.14 void DDS_QosProvider_return_publisher_qos (DDS_QosProvider * *qp,* DDS_PublisherQos * *qos*)

Return a PublisherQos back to the factory.

This will reclaim any resources allocated by the QosProvider.

1.2.2.15 void DDS_QosProvider_return_subscriber_qos (DDS_QosProvider * *qp,* DDS_SubscriberQos * *qos*)

Return a SubscriberQos back to the factory.

This will reclaim any resources allocated by the QosProvider.

1.2.2.16 void DDS_QosProvider_return_topic_qos (DDS_QosProvider * *qp,* DDS_TopicQos * *qos*)

Return a TopicQos back to the factory.

This will reclaim any resources allocated by the QosProvider.

1.3 DDS Conditions, Listeners, and WaitSets

Modules

- DDS Listeners
- DDS Conditions
- DDS WaitSets

1.3.1 Detailed Description

1.4 DDS Listeners

Data Structures

- 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.

1.4.1 Detailed Description

1.5 DDS Conditions

Data Structures

- 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.

1.5.1 Detailed Description

1.6 DDS WaitSets

Data Structures

- 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.*

1.6.1 Detailed Description

1.7 DDS Status Structures

Data Structures

- 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_QosPolicyCount

  Holds a DDS_QosPolicyId_t value and a counter to indicate the number of events associated with that PolicyId.
- 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.

1.7.1 Detailed Description

- DDS_InconsistentTopicStatus
- DDS_LivelinessChangedStatus
- DDS_LivelinessLostStatus
- DDS_OfferedDeadlineMissedStatus
- DDS_OfferedIncompatibleQosStatus
- DDS_PublicationMatchedStatus

- DDS_RequestedDeadlineMissedStatus
- DDS_RequestedIncompatibleQosStatus
- DDS_SampleLostStatus
- DDS_SampleRejectedStatus
- DDS_SubscriptionMatchedStatus

1.8 CoreDX DDS Transports

Data Structures

- struct CoreDX_UdpTransportConfig

  Structure that holds UDP Transport configuration items.
- struct CoreDX_TcpTransportConfig

  Structure that holds TCP Transport configuration items.
- struct CoreDX_LmtTransportConfig

  Structure that holds LMT Transport configuration items.
- struct CoreDX_SslTransportConfig

  Structure that holds SSL Transport configuration items.
- struct CoreDX_UdpTransport

  An instance of a Transport that can be added to a DomainParticipant.
- struct CoreDX_TcpTransport

  An instance of a Transport that can be added to a DomainParticipant.
- struct CoreDX_LmtTransport

  An instance of a Transport that can be added to a DomainParticipant.
- struct CoreDX_SslTransport

  An instance of a Transport that can be added to a DomainParticipant.

1.8.1 Detailed Description

1.9 X-Types DynamicType

Files

- file xtypes.h

  Provides the X-Types DDS_DynamicDataReader and DDS_DynamicDataWriter API.
- file xtypes_dtype.h

  Provides the X-Types DDS_DynamicType, DDS_DynamicData, and related APIs.

Data Structures

- struct DDS_AnnotationDescriptor

  A DDS_AnnotationDescriptor object comprises the state of an annotation as it is applied to some element.
- struct DDS_TypeDescriptor

  A DDS_TypeDescriptor comprises the state of a type.
- struct DDS_MemberDescriptor

  A DDS_MemberDescriptor comprises the state of a DDS_DynamicTypeMember.
- struct DDS_DynamicDataReader

  Provides a DataReader interface that is tailored to support reading an X-Types defined DynamicType data type. The specific DynamicType must have been registered previously with the DomainParticipant.
- struct DDS_DynamicDataWriter

  Provides a DataWriter interface that is tailored to support writing an X-Types defined DynamicType data type. The specific DynamicType must have been registered previously with the DomainParticipant.
- struct DDS_DynamicTypeBuilderFactory

  An instance of this type is responsible for creating DDS_DynamicType and DDS_DynamicTypeSupport objects.
- struct DDS_DynamicTypeBuilder

  A DDS_DynamicTypeBuilder object represents the state of a particular type defined according to the Type System. It is used to instantiate concrete DDS_DynamicType objects.
- struct DDS_DynamicTypeSupport

  The DDS_DynamicTypeSupport interface extends the DDS_TypeSupport interface defined by the DDS specification. This TypeSupport operates on DDS_DynamicData samples.
- struct DDS_DynamicType

  An instance of DDS_DynamicType represent a type's schema: its physical name, kind, member definitions (if any), and so on.
- struct DDS_DynamicTypeMember

  A DDS_DynamicTypeMember represents a "member" of a type. A "member" in this sense may be a member of an aggregated type, a constant within an enumeration, or some other type substructure.
- struct DDS_DynamicDataFactory

  This type is responsible for creating DDS_DynamicData instances.
- struct DDS_DynamicData

  A DDS_DynamicData object represents an individual data sample. It provides reflective getters and setters for the members of that sample.

Functions

- DDS_MAP_DECLARE (DDS_ObjectName, DDS_ObjectName, DDS_Parameters)

  A DDS_Parameters instance is a key-value map.
- DDS_MAP_DECLARE (char *, DDS_DynamicTypeMember, DDS_DynamicTypeMembersByName)

A DDS_DynamicTypeMembersByName instance is a map of DDS_DynamicTypeMember's indexed by 'name'.

- DDS_MAP_DECLARE (DDS_MemberId, DDS_DynamicTypeMember, DDS_DynamicTypeMembersById)

  A DDS_DynamicTypeMembersByName instance is a map of DDS_DynamicTypeMember's indexed by MemberId.
- DECLARE_SEQ (DDS_DynamicTypeMember, DDS_DynamicTypeMemberSeq)

  A DDS_DynamicTypeMemberSeq instance is a sequence of DDS_DynamicTypeMember's.
- DDS_DynamicTypeBuilderFactory DDS_DynamicTypeBuilderFactory_get_instance (void)

  Return the DDS_DynamicTypeBuilderFactory singleton instance.
- DDS_ReturnCode_t DDS_DynamicTypeBuilderFactory_delete_instance (void)

  Reclaim any resources associated with the instance returned from DDS_DynamicTypeBuilderFactory_get_instance.
- DDS_DynamicType DDS_DynamicTypeBuilderFactory_get_primitive_type (DDS_DynamicTypeBuilderFactory dtbf, const DDS_TypeKind kind)

  Retrieve a DDS_DynamicType object corresponding to the indicated primitive type kind.
- DDS_DynamicTypeBuilder DDS_DynamicTypeBuilderFactory_create_type (DDS_DynamicTypeBuilderFactory dtbf, const DDS_TypeDescriptor *descriptor)

  Create and return a new DDS_DynamicTypeBuilder object as described by the given type descriptor.
- DDS_DynamicTypeBuilder DDS_DynamicTypeBuilderFactory_create_type_copy (DDS_DynamicTypeBuilderFactory dtbf, const DDS_DynamicType type)

  Create and return a new DDS_DynamicTypeBuilder object with a copy of the state of the given type.
- DDS_DynamicTypeBuilder DDS_DynamicTypeBuilderFactory_create_type_w_type_object (DDS_DynamicTypeBuilderFactory dtbf, const DDS_TypeObject *type_object)

  Create and return a new DDS_DynamicTypeBuilder object that describes a type identical to that described by the given TypeObject.
- DDS_DynamicTypeBuilder DDS_DynamicTypeBuilderFactory_create_string_type (DDS_DynamicTypeBuilderFactory dtbf, const unsigned int bound)

  Create and return a new DDS_DynamicTypeBuilder object representing a string type.
- DDS_DynamicTypeBuilder DDS_DynamicTypeBuilderFactory_create_wstring_type (DDS_DynamicTypeBuilderFactory dtbf, const unsigned int bound)

  Create and return a new DDS_DynamicTypeBuilder object representing a string type.
- DDS_DynamicTypeBuilder DDS_DynamicTypeBuilderFactory_create_sequence_type (DDS_DynamicTypeBuilderFactory dtbf, const DDS_DynamicType element_type, const unsigned int bound)

  Create and return a new DDS_DynamicTypeBuilder object representing a sequence type.
- DDS_DynamicTypeBuilder DDS_DynamicTypeBuilderFactory_create_array_type (DDS_DynamicTypeBuilderFactory dtbf, const DDS_DynamicType element_type, const DDS_BoundSeq *bound)

  Create and return a new DDS_DynamicTypeBuilder object representing an array type.
- DDS_DynamicTypeBuilder DDS_DynamicTypeBuilderFactory_create_map_type (DDS_DynamicTypeBuilderFactory dtbf, const DDS_DynamicType key_element_type, const DDS_DynamicType element_type, const unsigned int bound)

  Create and return a new DDS_DynamicTypeBuilder object representing a map type.
- DDS_DynamicTypeBuilder DDS_DynamicTypeBuilderFactory_create_bitset_type (DDS_DynamicTypeBuilderFactory dtbf, const unsigned int bound)

  Create and return a new DDS_DynamicTypeBuilder object representing a bit set type.
- DDS_DynamicTypeBuilder DDS_DynamicTypeBuilderFactory_create_type_w_uri (DDS_DynamicTypeBuilderFactory dtbf, const char *document_url, const char *type_name, const DDS_IncludePathSeq *include_paths)

  Create and return a new DDS_DynamicTypeBuilder object by parsing the type description at the given URL.
- DDS_DynamicTypeBuilder DDS_DynamicTypeBuilderFactory_create_type_w_document (DDS_DynamicTypeBuilderFactory dtbf, const char *document, const char *type_name, const DDS_IncludePathSeq *include_paths)

  Create and return a new DDS_DynamicTypeBuilder object by parsing the type description contained in the given string.
- DDS_ReturnCode_t DDS_DynamicTypeBuilderFactory_delete_type_builder (DDS_DynamicTypeBuilderFactory dtbf, DDS_DynamicTypeBuilder type_builder)

Destroy a DDS_DynamicTypeBuilder instance obtained through one of the create methods on the DDS_DynamicTypeBuilderFactory.

- DDS_ReturnCode_t DDS_DynamicTypeBuilderFactory_delete_type (DDS_DynamicTypeBuilderFactory dtbf, DDS_DynamicType type)

  Destroy a DDS_DynamicType instance obtained through the DDS_DynamicTypeBuilderFactory_get_primitive_type operation.
- DDS_DynamicTypeBuilder DDS_DynamicTypeBuilderFactory_create_structure_type (DDS_DynamicTypeBuilderFactory dtbf, DDS_DynamicType base_type)

  Create and return a new DDS_DynamicTypeBuilder object representing a structure type.
- DDS_DynamicTypeBuilder DDS_DynamicTypeBuilderFactory_create_union_type (DDS_DynamicTypeBuilderFactory dtbf)

  Create and return a new DDS_DynamicTypeBuilder object representing a union type.
- DDS_DynamicTypeBuilder DDS_DynamicTypeBuilderFactory_create_alias_type (DDS_DynamicTypeBuilderFactory dtbf, const DDS_DynamicType base_type)

  Create and return a new DDS_DynamicTypeBuilder object representing an alias for the provided 'base_type'.
- DDS_DynamicTypeBuilder DDS_DynamicTypeBuilderFactory_create_enumeration_type (DDS_DynamicTypeBuilderFactory dtbf, const unsigned int bound)

  Create and return a new DDS_DynamicTypeBuilder object representing an enumeration type.
- DDS_ReturnCode_t DDS_DynamicTypeBuilderFactory_create_extensibility_annotation (DDS_DynamicTypeBuilderFactory dtbf, DDS_AnnotationDescriptor *ad, DDS_ExtensibilityKind ekind)

  Initialize an AnnotationDescriptor to represent an '@extensibility' annotation with the specified extensibility kind.
- DDS_ReturnCode_t DDS_DynamicTypeBuilderFactory_create_optional_annotation (DDS_DynamicTypeBuilderFactory dtbf, DDS_AnnotationDescriptor *ad)

  Initialize an AnnotationDescriptor to represent an '@optional' annotation.
- DDS_ReturnCode_t DDS_DynamicTypeBuilder_get_descriptor (DDS_DynamicTypeBuilder dtb, DDS_TypeDescriptor *descriptor)

  This operation provides a summary of the state of this type.
- const char * DDS_DynamicTypeBuilder_get_name (DDS_DynamicTypeBuilder dtb)

  This convenience operation provides the fully qualified name of this type. I.
- DDS_TypeKind DDS_DynamicTypeBuilder_get_kind (DDS_DynamicTypeBuilder dtb)

  This convenience operation indicates the kind of this type (e.g., integer, structure, etc.).
- DDS_ReturnCode_t DDS_DynamicTypeBuilder_get_member_by_name (DDS_DynamicTypeBuilder dtb, DDS_DynamicTypeMember *member, const DDS_ObjectName name)

  This operation accesses a member by name.
- DDS_ReturnCode_t DDS_DynamicTypeBuilder_get_all_members_by_name (DDS_DynamicTypeBuilder dtb, DDS_DynamicTypeMembersByName *member)

  This operation provides access to the 'members_by_name' map.
- DDS_ReturnCode_t DDS_DynamicTypeBuilder_get_member (DDS_DynamicTypeBuilder dtb, DDS_DynamicTypeMember *member, const DDS_MemberId id)

  This operation accesses a member by id.
- DDS_ReturnCode_t DDS_DynamicTypeBuilder_get_all_members (DDS_DynamicTypeBuilder dtb, DDS_DynamicTypeMembersBy *member)

  This operation provides access to the 'members_by_id' map.
- unsigned int DDS_DynamicTypeBuilder_get_annotation_count (DDS_DynamicTypeBuilder dtb)

  Return the number of annotations applied to this type.
- DDS_ReturnCode_t DDS_DynamicTypeBuilder_get_annotation (DDS_DynamicTypeBuilder dtb, DDS_AnnotationDescriptor *descriptor, const unsigned int idx)

  Access an annotation at the specified index. On success, the 'descriptor' parameter is set to the annotation.
- unsigned char DDS_DynamicTypeBuilder_equals (DDS_DynamicTypeBuilder dtb, const DDS_DynamicType other)

Two types shall be considered equal if and only if all of their respective properties are equal.

- DDS_ReturnCode_t DDS_DynamicTypeBuilder_add_member (DDS_DynamicTypeBuilder dtb, const DDS_MemberDescriptor *descriptor)

  Add a member to the type.
- DDS_ReturnCode_t DDS_DynamicTypeBuilder_apply_annotation (DDS_DynamicTypeBuilder dtb, const DDS_AnnotationDescriptor *descriptor)

  Apply the given annotation to this type.
- DDS_ReturnCode_t DDS_DynamicTypeBuilder_set_extensibility (DDS_DynamicTypeBuilder dtb, uint32_t ext)

  Assign the specified extensibility to the Builder (struct/union)
- DDS_DynamicType DDS_DynamicTypeBuilder_build (DDS_DynamicTypeBuilder dtb)

  Create an immutable DDS_DynamicType object defined by the builder's current state.
- DDS_ReturnCode_t DDS_DynamicTypeBuilder_delete_type (DDS_DynamicTypeBuilder dtb, DDS_DynamicType type)

  Destroy the type information contained in this DDS_DynamicTypeBuilder instance.
- DDS_DynamicTypeSupport DDS_DynamicTypeSupport_create_type_support (const DDS_DynamicType type)

  Create and return a new DDS_DynamicTypeSupport object capable of supporting the given type.
- DDS_ReturnCode_t DDS_DynamicTypeSupport_delete_type_support (DDS_DynamicTypeSupport type_support)

  Delete the given type support object, which was previously created by a call to DDS_DynamicTypeSupport_create_type_support.
- DDS_ReturnCode_t DDS_DynamicTypeSupport_register_type (DDS_DynamicTypeSupport dts, const DDS_DomainParticipant participant, const char *type_name)

  Registers the TypeSupport with the DDS_DomainParticipant.
- const char * DDS_DynamicTypeSupport_get_type_name (DDS_DynamicTypeSupport dts)

  Returns the name of the type supported by this TypeSupport instance.
- DDS_DynamicType DDS_DynamicTypeSupport_get_type (DDS_DynamicTypeSupport dts)

  Returns the DDS_DynamicType that is supported by this TypeSupport instance.
- DDS_ReturnCode_t DDS_DynamicType_get_descriptor (DDS_DynamicType dt, DDS_TypeDescriptor *descriptor)

  This operation provides a summary of the current state of this type.
- const char * DDS_DynamicType_get_name (DDS_DynamicType dt)

  This convenience operation provides the fully qualified name of this type.
- DDS_TypeKind DDS_DynamicType_get_kind (DDS_DynamicType dt)

  This convenience operation returns the kind of this type (e.g., integer, structure, etc.).
- DDS_ReturnCode_t DDS_DynamicType_get_member_by_name (DDS_DynamicType dt, DDS_DynamicTypeMember *member, const DDS_ObjectName name)

  This operation accesses a member by name.
- DDS_ReturnCode_t DDS_DynamicType_get_all_members_by_name (DDS_DynamicType dt, DDS_DynamicTypeMembersByNam *member)

  This operation provides access to the 'members_by_name' map.
- DDS_ReturnCode_t DDS_DynamicType_get_member (DDS_DynamicType dt, DDS_DynamicTypeMember *member, const DDS_MemberId id)

  This operation accesses a member by id.
- DDS_ReturnCode_t DDS_DynamicType_get_all_members (DDS_DynamicType dt, DDS_DynamicTypeMembersById *member)

  This operation provides access to the 'members_by_id' map.
- unsigned int DDS_DynamicType_get_annotation_count (DDS_DynamicType dt)

  Return the number of annotations applied to this type.
- DDS_ReturnCode_t DDS_DynamicType_get_annotation (DDS_DynamicType dt, DDS_AnnotationDescriptor *descriptor, const unsigned int idx)

Access an annotation at the specified index. On success, the 'descriptor' parameter is set to the annotation.

- unsigned char DDS_DynamicType_equals (DDS_DynamicType dt, const DDS_DynamicType other)

  Two types shall be considered equal if and only if all of their respective properties are equal.
- DDS_ReturnCode_t DDS_AnnotationDescriptor_get_value (const DDS_AnnotationDescriptor *ad, DDS_ObjectName *value, const DDS_ObjectName key)

  This accesses an annotation property 'value' associated with the 'key' name.
- DDS_ReturnCode_t DDS_AnnotationDescriptor_get_all_value (const DDS_AnnotationDescriptor *ad, DDS_Parameters *value)

  Access the map of all key-value pairs.
- DDS_ReturnCode_t DDS_AnnotationDescriptor_set_value (DDS_AnnotationDescriptor *ad, const DDS_ObjectName key, const DDS_ObjectName value)
- DDS_ReturnCode_t DDS_AnnotationDescriptor_copy_from (DDS_AnnotationDescriptor *ad, const DDS_AnnotationDescriptor *other)
- unsigned char DDS_AnnotationDescriptor_equals (DDS_AnnotationDescriptor *ad, const DDS_AnnotationDescriptor *other)
- unsigned char DDS_AnnotationDescriptor_is_consistent (const DDS_AnnotationDescriptor *ad)
- DDS_DynamicType DDS_AnnotationDescriptor_get_type (const DDS_AnnotationDescriptor *ad)
- DDS_ReturnCode_t DDS_TypeDescriptor_copy_from (DDS_TypeDescriptor *to, const DDS_TypeDescriptor *from)

  Overwrite the contents of this descriptor with those of another descriptor.
- unsigned char DDS_TypeDescriptor_equals (const DDS_TypeDescriptor *td, const DDS_TypeDescriptor *other)

  Compare two DDS_TypeDescriptor's.
- unsigned char DDS_TypeDescriptor_is_consistent (const DDS_TypeDescriptor *td)

  Indicates whether the states of all of this descriptor's properties are consistent.
- DDS_ReturnCode_t DDS_MemberDescriptor_copy_from (DDS_MemberDescriptor *to, const DDS_MemberDescriptor *from)

  Overwrite the contents of the 'to' descriptor with those of the provided 'from' descriptor.
- unsigned char DDS_MemberDescriptor_equals (DDS_MemberDescriptor *md, const DDS_MemberDescriptor *other)

  Compare two DDS_TypeDescriptor's.
- unsigned char DDS_MemberDescriptor_is_consistent (DDS_MemberDescriptor *md)

  Indicates whether the states of all of this descriptor's properties are consistent.
- DDS_ReturnCode_t DDS_DynamicTypeMember_get_descriptor (DDS_DynamicTypeMember dtm, DDS_MemberDescriptor *descriptor)

  This operation accesses the current state of this type.
- unsigned int DDS_DynamicTypeMember_get_annotation_count (DDS_DynamicTypeMember dtm)

  Return the number of annotations applied to this type.
- DDS_ReturnCode_t DDS_DynamicTypeMember_get_annotation (DDS_DynamicTypeMember dtm, DDS_AnnotationDescriptor *descriptor, const unsigned int idx)

  Access an annotation at the specified index. On success, the 'descriptor' parameter is set to the annotation.
- unsigned char DDS_DynamicTypeMember_equals (DDS_DynamicTypeMember dtm, const DDS_DynamicTypeMember other)

  Compare two DDS_DynamicTypeMember's.
- DDS_MemberId DDS_DynamicTypeMember_get_id (DDS_DynamicTypeMember dtm)

  This convenience operation provides the member ID of this member.
- const char * DDS_DynamicTypeMember_get_name (DDS_DynamicTypeMember dtm)

  This convenience operation provides the name of this member.
- DDS_MemberFlag DDS_DynamicTypeMember_get_flags (DDS_DynamicTypeMember dtm)

  Access the type flags.

- DDS_ReturnCode_t DDS_DynamicTypeMember_set_flags (DDS_DynamicTypeMember dtm, DDS_MemberFlag f)

  Set the type flags.
- DDS_DynamicDataFactory DDS_DynamicDataFactory_get_instance (void)

  Return a DDS_DynamicDataFactory instance that can be used to create and delete DDS_DynamicData instances.
- DDS_ReturnCode_t DDS_DynamicDataFactory_delete_instance (void)

  Reclaim any resources associated with the object(s) previously returned from get_instance.
- DDS_DynamicData DDS_DynamicDataFactory_create_data (DDS_DynamicDataFactory ddf, DDS_DynamicType dt)

  Create and return a new data sample. All objects returned by this operation should eventually be deleted by calling DDS_DynamicDataFactory_delete_data.
- DDS_ReturnCode_t DDS_DynamicDataFactory_delete_data (DDS_DynamicDataFactory ddf, DDS_DynamicData data)

  Dispose of a data sample, reclaiming any associated resources.
- DDS_ReturnCode_t DDS_DynamicData_get_type (DDS_DynamicData dd, DDS_DynamicType *dyn_type)

  This provides the type that defines the values within this sample.
- DDS_ReturnCode_t DDS_DynamicData_get_descriptor (DDS_DynamicData dd, DDS_MemberDescriptor *value, const DDS_MemberId id)

  This provides access to the descriptor for each value in this object, identified by the member ID.
- DDS_ReturnCode_t DDS_DynamicData_set_descriptor (DDS_DynamicData dd, const DDS_MemberId id, const DDS_MemberDescriptor *value)

  Set the descriptor of the member specified by 'id'.
- unsigned char DDS_DynamicData_equals (DDS_DynamicData dd, const DDS_DynamicData *other)

  Compare two DDS_DynamicData instances for equality.
- DDS_MemberId DDS_DynamicData_get_member_id_by_name (DDS_DynamicData dd, const DDS_ObjectName name)

  Access the 'id' of a member matching 'name'.
- DDS_MemberId DDS_DynamicData_get_member_id_at_index (DDS_DynamicData dd, const unsigned int index)

  Access the 'id' of a member at the specified index.
- unsigned int DDS_DynamicData_get_item_count (DDS_DynamicData dd)

  Access the "item count" of the DDS_DynamicData.
- DDS_ReturnCode_t DDS_DynamicData_clear_all_values (DDS_DynamicData dd)

  Clears all values from the instance.
- DDS_ReturnCode_t DDS_DynamicData_clear_nonkey_values (DDS_DynamicData dd)

  Clear all non-key values from the instance.
- DDS_ReturnCode_t DDS_DynamicData_clear_value (DDS_DynamicData dd, const DDS_MemberId id)

  Clear the specified value from the instance.
- DDS_DynamicData DDS_DynamicData_loan_value (DDS_DynamicData dd, const DDS_MemberId id)

  The "loan" operations loan to the application a DDS_DynamicData object representing a value within this sample.
- DDS_ReturnCode_t DDS_DynamicData_return_loaned_value (DDS_DynamicData dd, const DDS_DynamicData value)

  Return a "loaned" data sample.
- DDS_DynamicData DDS_DynamicData_clone (DDS_DynamicData dd)

  Create and return a new data sample with the same contents as this one.
- DDS_ReturnCode_t DDS_DynamicData_get_int32_value (DDS_DynamicData dd, int *value, const DDS_MemberId id)

  Access a 32bit integer value, identified by 'id', in the sample.

- DDS_ReturnCode_t DDS_DynamicData_set_int32_value (DDS_DynamicData dd, const DDS_MemberId id, const int value)

  Set a 32bit integer value, identified by 'id', in the sample.
- DDS_ReturnCode_t DDS_DynamicData_get_uint32_value (DDS_DynamicData dd, unsigned int *value, const DDS_MemberId id)

  Access an unsigned 32bit integer value, identified by 'id', in the sample.
- DDS_ReturnCode_t DDS_DynamicData_set_uint32_value (DDS_DynamicData dd, const DDS_MemberId id, const unsigned int value)

  Set an unsigned 32bit integer value, identified by 'id', in the sample.
- DDS_ReturnCode_t DDS_DynamicData_get_int16_value (DDS_DynamicData dd, short *value, const DDS_MemberId id)

  Access a 16bit integer value, identified by 'id', in the sample.
- DDS_ReturnCode_t DDS_DynamicData_set_int16_value (DDS_DynamicData dd, const DDS_MemberId id, const short value)

  Set a 16bit integer value, identified by 'id', in the sample.
- DDS_ReturnCode_t DDS_DynamicData_get_uint16_value (DDS_DynamicData dd, unsigned short *value, const DDS_MemberId id)

  Access an unsigned 16bit integer value, identified by 'id', in the sample.
- DDS_ReturnCode_t DDS_DynamicData_set_uint16_value (DDS_DynamicData dd, const DDS_MemberId id, const unsigned short value)

  Set an unsigned 16bit integer value, identified by 'id', in the sample.
- DDS_ReturnCode_t DDS_DynamicData_get_int64_value (DDS_DynamicData dd, int64_t *value, const DDS_MemberId id)

  Access a 64bit integer value, identified by 'id', in the sample.
- DDS_ReturnCode_t DDS_DynamicData_set_int64_value (DDS_DynamicData dd, const DDS_MemberId id, const int64_t value)

  Set a 64bit integer value, identified by 'id', in the sample.
- DDS_ReturnCode_t DDS_DynamicData_get_uint64_value (DDS_DynamicData dd, uint64_t *value, const DDS_MemberId id)

  Access an unsigned 64bit integer value, identified by 'id', in the sample.
- DDS_ReturnCode_t DDS_DynamicData_set_uint64_value (DDS_DynamicData dd, const DDS_MemberId id, const uint64_t value)

  Set an unsigned 64bit integer value, identified by 'id', in the sample.
- DDS_ReturnCode_t DDS_DynamicData_get_float32_value (DDS_DynamicData dd, float *value, const DDS_MemberId id)

  Access a float value, identified by 'id', in the sample.
- DDS_ReturnCode_t DDS_DynamicData_set_float32_value (DDS_DynamicData dd, const DDS_MemberId id, const float value)

  Set a float value, identified by 'id', in the sample.
- DDS_ReturnCode_t DDS_DynamicData_get_float64_value (DDS_DynamicData dd, double *value, const DDS_MemberId id)

  Access a double value, identified by 'id', in the sample.
- DDS_ReturnCode_t DDS_DynamicData_set_float64_value (DDS_DynamicData dd, const DDS_MemberId id, const double value)

  Set a double value, identified by 'id', in the sample.
- DDS_ReturnCode_t DDS_DynamicData_get_char8_value (DDS_DynamicData dd, char *value, const DDS_MemberId id)

  Access an 8bit character value, identified by 'id', in the sample.

- DDS_ReturnCode_t DDS_DynamicData_set_char8_value (DDS_DynamicData dd, const DDS_MemberId id, const char value)

  Set an 8bit character value, identified by 'id', in the sample.
- DDS_ReturnCode_t DDS_DynamicData_get_char32_value (DDS_DynamicData dd, cdx_char32_t *value, const DDS_MemberId id)

  Access a 32bit character value, identified by 'id', in the sample.
- DDS_ReturnCode_t DDS_DynamicData_set_char32_value (DDS_DynamicData dd, const DDS_MemberId id, const cdx_char32_t value)

  Set a 32bit character value, identified by 'id', in the sample.
- DDS_ReturnCode_t DDS_DynamicData_get_byte_value (DDS_DynamicData dd, unsigned char *value, const DDS_MemberId id)

  Access an 8bit value, identified by 'id', in the sample.
- DDS_ReturnCode_t DDS_DynamicData_set_byte_value (DDS_DynamicData dd, const DDS_MemberId id, const unsigned char value)

  Set an 8bit value, identified by 'id', in the sample.
- DDS_ReturnCode_t DDS_DynamicData_get_boolean_value (DDS_DynamicData dd, unsigned char *value, const DDS_MemberId id)

  Access a boolean value, identified by 'id', in the sample.
- DDS_ReturnCode_t DDS_DynamicData_set_boolean_value (DDS_DynamicData dd, const DDS_MemberId id, const unsigned char value)

  Set a boolean value, identified by 'id', in the sample.
- DDS_ReturnCode_t DDS_DynamicData_get_string_value (DDS_DynamicData dd, char **value, const DDS_MemberId id)

  Access a string value, identified by 'id', in the sample.
- DDS_ReturnCode_t DDS_DynamicData_set_string_value (DDS_DynamicData dd, const DDS_MemberId id, const char *value)

  Set a string value, identified by 'id', in the sample.
- DDS_ReturnCode_t DDS_DynamicData_get_wstring_value (DDS_DynamicData dd, cdx_char32_t **value, const DDS_MemberId id)

  Access a wstring value, identified by 'id', in the sample.
- DDS_ReturnCode_t DDS_DynamicData_set_wstring_value (DDS_DynamicData dd, const DDS_MemberId id, const cdx_char32_t *value)

  Set a wstring value, identified by 'id', in the sample.
- DDS_ReturnCode_t DDS_DynamicData_get_complex_value (DDS_DynamicData dd, DDS_DynamicData *value, const DDS_MemberId id)

  Access a complex value, identified by 'id', in the sample.
- DDS_ReturnCode_t DDS_DynamicData_set_complex_value (DDS_DynamicData dd, const DDS_MemberId id, const DDS_DynamicData value)

  Set a complex value, identified by 'id', in the sample.
- DDS_ReturnCode_t DDS_DynamicData_get_map_key_value (DDS_DynamicData dd, const char **key_str, const DDS_MemberId id)

  Access the key value of a map type, identified by 'id', in the sample.
- DDS_ReturnCode_t DDS_DynamicData_get_int32_values (DDS_DynamicData dd, DDS_Int32Seq *value, const DDS_MemberId id)

  Provide access to array values as a sequence. If 'dd' is an array type, then 'id' must be DDS_MEMBER_ID_INVALID. If 'dd' is an aggretage type, then 'id' must refer to a member that is of type array. The sequence is populated with a 'shallow' copy of the data. This means that modifications to any of the values in the sequence will result in modifications to the data in the DDS_DynamicData instance. It is an error to 'clear' the returned sequence, as this will destroy memory that the DynamicData owns.

- DDS_ReturnCode_t DDS_DynamicData_set_int32_values (DDS_DynamicData dd, const DDS_MemberId id, const DDS_Int32Seq *value)

  Modify an array of values in the DDS_DynamicData instance.
- DDS_ReturnCode_t DDS_DynamicData_get_uint32_values (DDS_DynamicData dd, DDS_UInt32Seq *value, const DDS_MemberId id)

  Provide access to array values as a sequence. If 'dd' is an array type, then 'id' must be DDS_MEMBER_ID_INVALID. If 'dd' is an aggretage type, then 'id' must refer to a member that is of type array. The sequence is populated with a 'shallow' copy of the data. This means that modifications to any of the values in the sequence will result in modifications to the data in the DDS_DynamicData instance. It is an error to 'clear' the returned sequence, as this will destroy memory that the DynamicData owns.
- DDS_ReturnCode_t DDS_DynamicData_set_uint32_values (DDS_DynamicData dd, const DDS_MemberId id, const DDS_UInt32Seq *value)

  Modify an array of values in the DDS_DynamicData instance. If 'dd' is an array type, then 'id' must be DDS_MEMBER_ID_INVALID. If 'dd' is an aggretage type, then 'id' must refer to a member that is of type array. This copies the data into the DynamicData instance 'dd'. Modifications to the value sequence after this call will not have an effect on the contents of 'dd'.
- DDS_ReturnCode_t DDS_DynamicData_get_int16_values (DDS_DynamicData dd, DDS_Int16Seq *value, const DDS_MemberId id)

  Provide access to array values as a sequence. If 'dd' is an array type, then 'id' must be DDS_MEMBER_ID_INVALID. If 'dd' is an aggretage type, then 'id' must refer to a member that is of type array. The sequence is populated with a 'shallow' copy of the data. This means that modifications to any of the values in the sequence will result in modifications to the data in the DDS_DynamicData instance. It is an error to 'clear' the returned sequence, as this will destroy memory that the DynamicData owns.
- DDS_ReturnCode_t DDS_DynamicData_set_int16_values (DDS_DynamicData dd, const DDS_MemberId id, const DDS_Int16Seq *value)

  Modify an array of values in the DDS_DynamicData instance. If 'dd' is an array type, then 'id' must be DDS_MEMBER_ID_INVALID. If 'dd' is an aggretage type, then 'id' must refer to a member that is of type array. This copies the data into the DynamicData instance 'dd'. Modifications to the value sequence after this call will not have an effect on the contents of 'dd'.
- DDS_ReturnCode_t DDS_DynamicData_get_uint16_values (DDS_DynamicData dd, DDS_UInt16Seq *value, const DDS_MemberId id)

  Provide access to array values as a sequence. If 'dd' is an array type, then 'id' must be DDS_MEMBER_ID_INVALID. If 'dd' is an aggretage type, then 'id' must refer to a member that is of type array. The sequence is populated with a 'shallow' copy of the data. This means that modifications to any of the values in the sequence will result in modifications to the data in the DDS_DynamicData instance. It is an error to 'clear' the returned sequence, as this will destroy memory that the DynamicData owns.
- DDS_ReturnCode_t DDS_DynamicData_set_uint16_values (DDS_DynamicData dd, const DDS_MemberId id, const DDS_UInt16Seq *value)

  Modify an array of values in the DDS_DynamicData instance. If 'dd' is an array type, then 'id' must be DDS_MEMBER_ID_INVALID. If 'dd' is an aggretage type, then 'id' must refer to a member that is of type array. This copies the data into the DynamicData instance 'dd'. Modifications to the value sequence after this call will not have an effect on the contents of 'dd'.
- DDS_ReturnCode_t DDS_DynamicData_get_int64_values (DDS_DynamicData dd, DDS_Int64Seq *value, const DDS_MemberId id)

  Provide access to array values as a sequence. If 'dd' is an array type, then 'id' must be DDS_MEMBER_ID_INVALID. If 'dd' is an aggretage type, then 'id' must refer to a member that is of type array. The sequence is populated with a 'shallow' copy of the data. This means that modifications to any of the values in the sequence will result in modifications to the data in the DDS_DynamicData instance. It is an error to 'clear' the returned sequence, as this will destroy memory that the DynamicData owns.
- DDS_ReturnCode_t DDS_DynamicData_set_int64_values (DDS_DynamicData dd, const DDS_MemberId id, const DDS_Int64Seq *value)

  Modify an array of values in the DDS_DynamicData instance. If 'dd' is an array type, then 'id' must be DDS_MEMBER_ID_INVALID. If 'dd' is an aggretage type, then 'id' must refer to a member that is of type array. This copies the data into the DynamicData instance 'dd'. Modifications to the value sequence after this call will not have an effect on the contents of 'dd'.

- DDS_ReturnCode_t DDS_DynamicData_get_uint64_values (DDS_DynamicData dd, DDS_UInt64Seq *value, const DDS_MemberId id)

  Provide access to array values as a sequence. If 'dd' is an array type, then 'id' must be DDS_MEMBER_ID_INVALID. If 'dd' is an aggretage type, then 'id' must refer to a member that is of type array. The sequence is populated with a 'shallow' copy of the data. This means that modifications to any of the values in the sequence will result in modifications to the data in the DDS_DynamicData instance. It is an error to 'clear' the returned sequence, as this will destroy memory that the DynamicData owns.
- DDS_ReturnCode_t DDS_DynamicData_set_uint64_values (DDS_DynamicData dd, const DDS_MemberId id, const DDS_UInt64Seq *value)

  Modify an array of values in the DDS_DynamicData instance. If 'dd' is an array type, then 'id' must be DDS_MEMBER_ID_INVALID. If 'dd' is an aggretage type, then 'id' must refer to a member that is of type array. This copies the data into the DynamicData instance 'dd'. Modifications to the value sequence after this call will not have an effect on the contents of 'dd'.
- DDS_ReturnCode_t DDS_DynamicData_get_float32_values (DDS_DynamicData dd, DDS_Float32Seq *value, const DDS_MemberId id)

  Provide access to array values as a sequence. If 'dd' is an array type, then 'id' must be DDS_MEMBER_ID_INVALID. If 'dd' is an aggretage type, then 'id' must refer to a member that is of type array. The sequence is populated with a 'shallow' copy of the data. This means that modifications to any of the values in the sequence will result in modifications to the data in the DDS_DynamicData instance. It is an error to 'clear' the returned sequence, as this will destroy memory that the DynamicData owns.
- DDS_ReturnCode_t DDS_DynamicData_set_float32_values (DDS_DynamicData dd, const DDS_MemberId id, const DDS_Float32Seq *value)

  Modify an array of values in the DDS_DynamicData instance. If 'dd' is an array type, then 'id' must be DDS_MEMBER_ID_INVALID. If 'dd' is an aggretage type, then 'id' must refer to a member that is of type array. This copies the data into the DynamicData instance 'dd'. Modifications to the value sequence after this call will not have an effect on the contents of 'dd'.
- DDS_ReturnCode_t DDS_DynamicData_get_float64_values (DDS_DynamicData dd, DDS_Float64Seq *value, const DDS_MemberId id)

  Provide access to array values as a sequence. If 'dd' is an array type, then 'id' must be DDS_MEMBER_ID_INVALID. If 'dd' is an aggretage type, then 'id' must refer to a member that is of type array. The sequence is populated with a 'shallow' copy of the data. This means that modifications to any of the values in the sequence will result in modifications to the data in the DDS_DynamicData instance. It is an error to 'clear' the returned sequence, as this will destroy memory that the DynamicData owns.
- DDS_ReturnCode_t DDS_DynamicData_set_float64_values (DDS_DynamicData dd, const DDS_MemberId id, const DDS_Float64Seq *value)

  Modify an array of values in the DDS_DynamicData instance. If 'dd' is an array type, then 'id' must be DDS_MEMBER_ID_INVALID. If 'dd' is an aggretage type, then 'id' must refer to a member that is of type array. This copies the data into the DynamicData instance 'dd'. Modifications to the value sequence after this call will not have an effect on the contents of 'dd'.
- DDS_ReturnCode_t DDS_DynamicData_get_char8_values (DDS_DynamicData dd, DDS_CharSeq *value, const DDS_MemberId id)

  Provide access to array values as a sequence. If 'dd' is an array type, then 'id' must be DDS_MEMBER_ID_INVALID. If 'dd' is an aggretage type, then 'id' must refer to a member that is of type array. The sequence is populated with a 'shallow' copy of the data. This means that modifications to any of the values in the sequence will result in modifications to the data in the DDS_DynamicData instance. It is an error to 'clear' the returned sequence, as this will destroy memory that the DynamicData owns.
- DDS_ReturnCode_t DDS_DynamicData_set_char8_values (DDS_DynamicData dd, const DDS_MemberId id, const DDS_CharSeq *value)

  Modify an array of values in the DDS_DynamicData instance. If 'dd' is an array type, then 'id' must be DDS_MEMBER_ID_INVALID. If 'dd' is an aggretage type, then 'id' must refer to a member that is of type array. This copies the data into the DynamicData instance 'dd'. Modifications to the value sequence after this call will not have an effect on the contents of 'dd'.
- DDS_ReturnCode_t DDS_DynamicData_get_char32_values (DDS_DynamicData dd, DDS_WcharSeq *value, const DDS_MemberId id)

Provide access to array values as a sequence. If 'dd' is an array type, then 'id' must be DDS_MEMBER_ID_INVALID. If 'dd' is an aggretage type, then 'id' must refer to a member that is of type array. The sequence is populated with a 'shallow' copy of the data. This means that modifications to any of the values in the sequence will result in modifications to the data in the DDS_DynamicData instance. It is an error to 'clear' the returned sequence, as this will destroy memory that the DynamicData owns.

- DDS_ReturnCode_t DDS_DynamicData_set_char32_values (DDS_DynamicData dd, const DDS_MemberId id, const DDS_WcharSeq *value)

  Modify an array of values in the DDS_DynamicData instance. If 'dd' is an array type, then 'id' must be DDS_MEMBER_ID_INVALID. If 'dd' is an aggretage type, then 'id' must refer to a member that is of type array. This copies the data into the DynamicData instance 'dd'. Modifications to the value sequence after this call will not have an effect on the contents of 'dd'.

- DDS_ReturnCode_t DDS_DynamicData_get_byte_values (DDS_DynamicData dd, DDS_ByteSeq *value, const DDS_MemberId id)

  Provide access to array values as a sequence. If 'dd' is an array type, then 'id' must be DDS_MEMBER_ID_INVALID. If 'dd' is an aggretage type, then 'id' must refer to a member that is of type array. The sequence is populated with a 'shallow' copy of the data. This means that modifications to any of the values in the sequence will result in modifications to the data in the DDS_DynamicData instance. It is an error to 'clear' the returned sequence, as this will destroy memory that the DynamicData owns.

- DDS_ReturnCode_t DDS_DynamicData_set_byte_values (DDS_DynamicData dd, const DDS_MemberId id, const DDS_ByteSeq *value)

  Modify an array of values in the DDS_DynamicData instance. If 'dd' is an array type, then 'id' must be DDS_MEMBER_ID_INVALID. If 'dd' is an aggretage type, then 'id' must refer to a member that is of type array. This copies the data into the DynamicData instance 'dd'. Modifications to the value sequence after this call will not have an effect on the contents of 'dd'.

- DDS_ReturnCode_t DDS_DynamicData_get_boolean_values (DDS_DynamicData dd, DDS_BooleanSeq *value, const DDS_MemberId id)

  Provide access to array values as a sequence. If 'dd' is an array type, then 'id' must be DDS_MEMBER_ID_INVALID. If 'dd' is an aggretage type, then 'id' must refer to a member that is of type array. The sequence is populated with a 'shallow' copy of the data. This means that modifications to any of the values in the sequence will result in modifications to the data in the DDS_DynamicData instance. It is an error to 'clear' the returned sequence, as this will destroy memory that the DynamicData owns.

- DDS_ReturnCode_t DDS_DynamicData_set_boolean_values (DDS_DynamicData dd, const DDS_MemberId id, const DDS_BooleanSeq *value)

  Modify an array of values in the DDS_DynamicData instance. If 'dd' is an array type, then 'id' must be DDS_MEMBER_ID_INVALID. If 'dd' is an aggretage type, then 'id' must refer to a member that is of type array. This copies the data into the DynamicData instance 'dd'. Modifications to the value sequence after this call will not have an effect on the contents of 'dd'.

- DDS_ReturnCode_t DDS_DynamicData_get_string_values (DDS_DynamicData dd, DDS_StringSeq *value, const DDS_MemberId id)

  Provide access to array values as a sequence. If 'dd' is an array type, then 'id' must be DDS_MEMBER_ID_INVALID. If 'dd' is an aggretage type, then 'id' must refer to a member that is of type array. The sequence is populated with a 'shallow' copy of the data. This means that modifications to any of the values in the sequence will result in modifications to the data in the DDS_DynamicData instance. It is an error to 'clear' the returned sequence, as this will destroy memory that the DynamicData owns.

- DDS_ReturnCode_t DDS_DynamicData_set_string_values (DDS_DynamicData dd, const DDS_MemberId id, const DDS_StringSeq *value)

  Modify an array of values in the DDS_DynamicData instance. If 'dd' is an array type, then 'id' must be DDS_MEMBER_ID_INVALID. If 'dd' is an aggretage type, then 'id' must refer to a member that is of type array. This copies the data into the DynamicData instance 'dd'. Modifications to the value sequence after this call will not have an effect on the contents of 'dd'.

- DDS_ReturnCode_t DDS_DynamicData_get_wstring_values (DDS_DynamicData dd, DDS_WstringSeq *value, const DDS_MemberId id)

  Provide access to array values as a sequence. If 'dd' is an array type, then 'id' must be DDS_MEMBER_ID_INVALID. If 'dd' is an aggretage type, then 'id' must refer to a member that is of type array. The sequence is populated with a 'shallow'

copy of the data. This means that modifications to any of the values in the sequence will result in modifications to the data in the DDS_DynamicData instance. It is an error to 'clear' the returned sequence, as this will destroy memory that the DynamicData owns.

- DDS_ReturnCode_t DDS_DynamicData_set_wstring_values (DDS_DynamicData dd, const DDS_MemberId id, const DDS_WstringSeq *value)

  Modify an array of values in the DDS_DynamicData instance. If 'dd' is an array type, then 'id' must be DDS_MEMBER_ID_INVALID. If 'dd' is an aggretage type, then 'id' must refer to a member that is of type array. This copies the data into the DynamicData instance 'dd'. Modifications to the value sequence after this call will not have an effect on the contents of 'dd'.

- DDS_ReturnCode_t DDS_DynamicDataFactory_create_value (DDS_DynamicDataFactory ddf, const DDS_DynamicData data, const DDS_MemberId id, DDS_DynamicData *value)

  Creates a DynamicData instance for the specified member (id) of 'data'. This is useful for 'optional' members that are not constructed by the DDS_DynamicData_create_data() call. By default 'optional' members are left in the 'null' state. To set them, call create_value() and then set_complex_value().

1.9.1 Detailed Description

1.9.2 Function Documentation

1.9.2.1 DDS_ReturnCode_t DDS_AnnotationDescriptor_copy_from (DDS_AnnotationDescriptor * *ad*, const DDS_AnnotationDescriptor * *other*) [related]

Not Yet Supported This operation is not yet implemented.

1.9.2.2 unsigned char DDS_AnnotationDescriptor_equals (DDS_AnnotationDescriptor * *ad*, const DDS_AnnotationDescriptor * *other*) [related]

Not Yet Supported This operation is not yet implemented.

1.9.2.3 DDS_ReturnCode_t DDS_AnnotationDescriptor_get_all_value (const DDS_AnnotationDescriptor * *ad*, DDS_Parameters * *value*) [related]

Access the map of all key-value pairs.

Return values

DDS_RETCODE_OK	on success, the 'value' parameters map is populated.

Not Yet Supported This operation is not yet implemented.

1.9.2.4 DDS_DynamicType DDS_AnnotationDescriptor_get_type (const DDS_AnnotationDescriptor * *ad*) [related]

Not Yet Supported This operation is not yet implemented.

1.9.2.5 DDS_ReturnCode_t DDS_AnnotationDescriptor_get_value (const DDS_AnnotationDescriptor * *ad*, DDS_ObjectName * *value*, const DDS_ObjectName *key*) [related]

This accesses an annotation property 'value' associated with the 'key' name.

Every attribute value is represented as a string although annotation type members can have other types as well. A string representation of a data value is considered well formed if it would be a valid IDL literal of the corresponding type with the following qualifications:

- String and character literals shall not be surrounded by quotation characters (‘"’ or ‘'’)
- All expressions shall be fully evaluated such that no operators or other non-literal characters occur in the value. For example, “5” shall be considered a well-formed string representation of the integer quantity five, but “2 + ENUM_VALUE_THREE” shall not be.

Not Yet Supported This operation is not yet implemented.

1.9.2.6 unsigned char DDS_AnnotationDescriptor_is_consistent (const DDS_AnnotationDescriptor * *ad*) [related]

Not Yet Supported This operation is not yet implemented.

1.9.2.7 DDS_ReturnCode_t DDS_AnnotationDescriptor_set_value (DDS_AnnotationDescriptor * *ad*, const DDS_ObjectName *key*, const DDS_ObjectName *value*) [related]

Not Yet Supported This operation is not yet implemented.

1.9.2.8 DDS_ReturnCode_t DDS_DynamicData_clear_all_values (DDS_DynamicData *dd*) [related]

Clears all values from the instance.

The meaning of "clearing" a member depends on the type of data represented by this sample:

- If this sample is of an aggregated type, and the indicated member is optional, remove it. If the indicated member is not optional, set it to its default value.
- If this sample is of a variable-length collection type, remove the indicated element, shifting any subsequent elements to the next-lowest index.
- If the sample is of an array type, set the indicated element to its default value.
- If the sample is of a bit set type, clear the indicated bit.
- If the sample is of an enumerated type, set it to the first value of the enumerated type.
- If the sample is of a primitive type, set it to its default value.
- The clear_all_members takes the above action for each value in turn.
- The clear_nonkey_value operation has exactly the same effect as clear_all_values with one exception: the values of key fields of aggregated types retain their values.

1.9.2.9 DDS_ReturnCode_t DDS_DynamicData_clear_nonkey_values (DDS_DynamicData *dd*) [related]

Clear all non-key values from the instance.

See also

DDS_DynamicData_clear_all_values

1.9.2.10 DDS_ReturnCode_t DDS_DynamicData_clear_value (DDS_DynamicData *dd,* const DDS_MemberId *id*) [related]

Clear the specified value from the instance.

See also

DDS_DynamicData_clear_all_values

1.9.2.11 DDS_DynamicData DDS_DynamicData_clone (DDS_DynamicData *dd*) [related]

Create and return a new data sample with the same contents as this one.

A comparison of this object and the clone using equals immediately following this call will return true.

1.9.2.12 unsigned char DDS_DynamicData_equals (DDS_DynamicData *dd,* const DDS_DynamicData * *other*) [related]

Compare two DDS_DynamicData instances for equality.

Two data samples are considered to be equal if and only if all of the following conditions hold:

- Their respective type definitions are equal.
- All contained values are equal and occur in the same order.

If the samples' type is an aggregated type, the previous rule is amended as follows:

- Members shall be compared without regard to their order.
- One of the samples may omit a non-optional member that is present in the other if that member takes its default value in the latter sample.

1.9.2.13 DDS_ReturnCode_t DDS_DynamicData_get_boolean_value (DDS_DynamicData *dd,* unsigned char * *value,* const DDS_MemberId *id*) [related]

Access a boolean value, identified by 'id', in the sample.

If 'id' is DDS_MEMBER_ID_INVALID, then it attempts to treat 'dd' as the requested type. Otherwise, it looks in 'dd' for a value that matches that 'id'. As described above, 'id' has different meanings based on the type of 'dd'.

Return values

DDS_RETCODE_BAD_PARAMETER	in the case of an invalid parameter or non-existent 'id'.

1.9.2.14 DDS_ReturnCode_t DDS_DynamicData_get_byte_value (DDS_DynamicData *dd,* unsigned char * *value,* const DDS_MemberId *id*) [related]

Access an 8bit value, identified by 'id', in the sample.

If 'id' is DDS_MEMBER_ID_INVALID, then it attempts to treat 'dd' as the requested type. Otherwise, it looks in 'dd' for a value that matches that 'id'. As described above, 'id' has different meanings based on the type of 'dd'.

Return values

DDS_RETCODE_BAD_PARAMETER	in the case of an invalid parameter or non-existent 'id'.

1.9.2.15 DDS_ReturnCode_t DDS_DynamicData_get_char32_value (DDS_DynamicData *dd,* cdx_char32_t * *value,* const DDS_MemberId *id*) [related]

Access a 32bit character value, identified by 'id', in the sample.

If 'id' is DDS_MEMBER_ID_INVALID, then it attempts to treat 'dd' as the requested type. Otherwise, it looks in 'dd' for a value that matches that 'id'. As described above, 'id' has different meanings based on the type of 'dd'.

Return values

DDS_RETCODE_BAD_PARAMETER	in the case of an invalid parameter or non-existent 'id'.

1.9.2.16 DDS_ReturnCode_t DDS_DynamicData_get_char8_value (DDS_DynamicData *dd,* char * *value,* const DDS_MemberId *id*) [related]

Access an 8bit character value, identified by 'id', in the sample.

If 'id' is DDS_MEMBER_ID_INVALID, then it attempts to treat 'dd' as the requested type. Otherwise, it looks in 'dd' for a value that matches that 'id'. As described above, 'id' has different meanings based on the type of 'dd'.

Return values

DDS_RETCODE_BAD_PARAMETER	in the case of an invalid parameter or non-existent 'id'.

1.9.2.17 DDS_ReturnCode_t DDS_DynamicData_get_complex_value (DDS_DynamicData *dd,* DDS_DynamicData * *value,* const DDS_MemberId *id*) [related]

Access a complex value, identified by 'id', in the sample.

If 'id' is DDS_MEMBER_ID_INVALID, then it attempts to treat 'dd' as the requested type. Otherwise, it looks in 'dd' for a value that matches that 'id'. As described above, 'id' has different meanings based on the type of 'dd'.

Return values

DDS_RETCODE_BAD_PARAMETER	in the case of an invalid parameter or non-existent 'id'.

1.9.2.18 DDS_ReturnCode_t DDS_DynamicData_get_descriptor (DDS_DynamicData *dd*, DDS_MemberDescriptor * *value*, const DDS_MemberId *id*) [related]

This provides access to the descriptor for each value in this object, identified by the member ID.

See the description of DDS_DynamicData values.

1.9.2.19 DDS_ReturnCode_t DDS_DynamicData_get_float32_value (DDS_DynamicData *dd*, float * *value*, const DDS_MemberId *id*) [related]

Access a float value, identified by 'id', in the sample.

If 'id' is DDS_MEMBER_ID_INVALID, then it attempts to treat 'dd' as the requested type. Otherwise, it looks in 'dd' for a value that matches that 'id'. As described above, 'id' has different meanings based on the type of 'dd'.

Return values

DDS_RETCODE_BAD_PARAMETER	in the case of an invalid parameter or non-existent 'id'.

1.9.2.20 DDS_ReturnCode_t DDS_DynamicData_get_float64_value (DDS_DynamicData *dd*, double * *value*, const DDS_MemberId *id*) [related]

Access a double value, identified by 'id', in the sample.

If 'id' is DDS_MEMBER_ID_INVALID, then it attempts to treat 'dd' as the requested type. Otherwise, it looks in 'dd' for a value that matches that 'id'. As described above, 'id' has different meanings based on the type of 'dd'.

Return values

DDS_RETCODE_BAD_PARAMETER	in the case of an invalid parameter or non-existent 'id'.

1.9.2.21 DDS_ReturnCode_t DDS_DynamicData_get_int16_value (DDS_DynamicData *dd*, short * *value*, const DDS_MemberId *id*) [related]

Access a 16bit integer value, identified by 'id', in the sample.

If 'id' is DDS_MEMBER_ID_INVALID, then it attempts to treat 'dd' as the requested type. Otherwise, it looks in 'dd' for a value that matches that 'id'. As described above, 'id' has different meanings based on the type of 'dd'.

Return values

DDS_RETCODE_BAD_PARAMETER	in the case of an invalid parameter or non-existent 'id'.

1.9.2.22 DDS_ReturnCode_t DDS_DynamicData_get_int32_value (DDS_DynamicData *dd*, int * *value*, const DDS_MemberId *id*) [related]

Access a 32bit integer value, identified by 'id', in the sample.

If 'id' is DDS_MEMBER_ID_INVALID, then it attempts to treat 'dd' as the requested type. Otherwise, it looks in 'dd' for a value that matches that 'id'. As described above, 'id' has different meanings based on the type of 'dd'.

Return values

DDS_RETCODE_BAD_PARAMETER	in the case of an invalid parameter or non-existent 'id'.

1.9.2.23 DDS_ReturnCode_t DDS_DynamicData_get_int64_value (DDS_DynamicData *dd*, int64_t * *value*, const DDS_MemberId *id*) [related]

Access a 64bit integer value, identified by 'id', in the sample.

If 'id' is DDS_MEMBER_ID_INVALID, then it attempts to treat 'dd' as the requested type. Otherwise, it looks in 'dd' for a value that matches that 'id'. As described above, 'id' has different meanings based on the type of 'dd'.

Return values

DDS_RETCODE_BAD_PARAMETER	in the case of an invalid parameter or non-existent 'id'.

1.9.2.24 unsigned int DDS_DynamicData_get_item_count (DDS_DynamicData *dd*) [related]

Access the "item count" of the DDS_DynamicData.

The "item count" of the data depends on the type of the object.

- If the object is of a collection type, return the number of elements currently in the collection.
- In the case of an array type, this value will always be equal to the product of the bounds of all array dimensions.
- If the object is of a bit set type, return the number of named flags that are currently set in the bit set.
- If the object is of a structure or annotation type, return the number of members in the object. This value may be different than the number of members in the corresponding DDS_DynamicType; for example, some optional members may be omitted.
- If the object is of a union type, return the number of members in the object. This value will always be two: the discriminator and the current member corresponding to it.
- If the object is of a primitive or enumeration type, it is atomic, and the count is one.
- If the object is of an alias type, return the value appropriate for the alias's base type.

1.9.2.25 DDS_ReturnCode_t DDS_DynamicData_get_map_key_value (DDS_DynamicData *dd*, const char ** *key_str*, const DDS_MemberId *id*) [related]

Access the key value of a map type, identified by 'id', in the sample.

This treats 'dd' as a map, and returns the key value of the pair at index 'id'.

Return values

DDS_RETCODE_BAD_PARAMETER	in the case of an invalid parameter or out of range 'id'.

1.9.2.26 DDS_ReturnCode_t DDS_DynamicData_get_string_value (DDS_DynamicData *dd,* char ** *value,* const DDS_MemberId *id*) [related]

Access a string value, identified by 'id', in the sample.

If 'id' is DDS_MEMBER_ID_INVALID, then it attempts to treat 'dd' as the requested type. Otherwise, it looks in 'dd' for a value that matches that 'id'. As described above, 'id' has different meanings based on the type of 'dd'.

Return values

DDS_RETCODE_BAD_PARAMETER	in the case of an invalid parameter or non-existent 'id'.

1.9.2.27 DDS_ReturnCode_t DDS_DynamicData_get_uint16_value (DDS_DynamicData *dd,* unsigned short * *value,* const DDS_MemberId *id*) [related]

Access an unsigned 16bit integer value, identified by 'id', in the sample.

If 'id' is DDS_MEMBER_ID_INVALID, then it attempts to treat 'dd' as the requested type. Otherwise, it looks in 'dd' for a value that matches that 'id'. As described above, 'id' has different meanings based on the type of 'dd'.

Return values

DDS_RETCODE_BAD_PARAMETER	in the case of an invalid parameter or non-existent 'id'.

1.9.2.28 DDS_ReturnCode_t DDS_DynamicData_get_uint32_value (DDS_DynamicData *dd,* unsigned int * *value,* const DDS_MemberId *id*) [related]

Access an unsigned 32bit integer value, identified by 'id', in the sample.

If 'id' is DDS_MEMBER_ID_INVALID, then it attempts to treat 'dd' as the requested type. Otherwise, it looks in 'dd' for a value that matches that 'id'. As described above, 'id' has different meanings based on the type of 'dd'.

Return values

DDS_RETCODE_BAD_PARAMETER	in the case of an invalid parameter or non-existent 'id'.

1.9.2.29 DDS_ReturnCode_t DDS_DynamicData_get_uint64_value (DDS_DynamicData *dd,* uint64_t * *value,* const DDS_MemberId *id*) [related]

Access an unsigned 64bit integer value, identified by 'id', in the sample.

If 'id' is DDS_MEMBER_ID_INVALID, then it attempts to treat 'dd' as the requested type. Otherwise, it looks in 'dd' for a value that matches that 'id'. As described above, 'id' has different meanings based on the type of 'dd'.

Return values

DDS_RETCODE_BAD_PARAMETER	in the case of an invalid parameter or non-existent 'id'.

1.9.2.30 DDS_ReturnCode_t DDS_DynamicData_get_wstring_value (DDS_DynamicData *dd*, cdx_char32_t ** *value*, const DDS_MemberId *id*) [related]

Access a wstring value, identified by 'id', in the sample.

If 'id' is DDS_MEMBER_ID_INVALID, then it attempts to treat 'dd' as the requested type. Otherwise, it looks in 'dd' for a value that matches that 'id'. As described above, 'id' has different meanings based on the type of 'dd'.

Return values

DDS_RETCODE_BAD_PARAMETER	in the case of an invalid parameter or non-existent 'id'.

1.9.2.31 DDS_DynamicData DDS_DynamicData_loan_value (DDS_DynamicData *dd*, const DDS_MemberId *id*) [related]

The "loan" operations loan to the application a DDS_DynamicData object representing a value within this sample.

This operation allows applications to visit values without allocating additional DDS_DynamicData objects or copying values. This loan shall be returned by the DDS_DynamicData_return_loaned_value operation.

A given DDS_DynamicData object may support only a single outstanding loan at a time. That is, after calling a "loan" operation, an application must subsequently call return_loaned_value before calling a loan operation again. If an application violates this constraint, the loan operation shall return a nil value.

A loan operation shall also return a nil value if the indicated value does not exist.

1.9.2.32 DDS_ReturnCode_t DDS_DynamicData_return_loaned_value (DDS_DynamicData *dd*, const DDS_DynamicData *value*) [related]

Return a "loaned" data sample.

The DDS_DynamicData_return_loaned_value operation may return DDS_RETCODE_PRECONDITION_NOT_MET if the provided sample object does not represent an outstanding loan from the sample on which the operation is invoked.

1.9.2.33 DDS_ReturnCode_t DDS_DynamicData_set_boolean_value (DDS_DynamicData *dd*, const DDS_MemberId *id*, const unsigned char *value*) [related]

Set a boolean value, identified by 'id', in the sample.

If 'id' is DDS_MEMBER_ID_INVALID, then it attempts to treat 'dd' as the requested type. Otherwise, it looks in 'dd' for a value that matches that 'id'. As described above, 'id' has different meanings based on the type of 'dd'.

Return values

DDS_RETCODE_BAD_PARAMETER	in the case of an invalid parameter or non-existent 'id'.

1.9.2.34 DDS_ReturnCode_t DDS_DynamicData_set_byte_value (DDS_DynamicData *dd*, const DDS_MemberId *id*, const unsigned char *value*) [related]

Set an 8bit value, identified by 'id', in the sample.

If 'id' is DDS_MEMBER_ID_INVALID, then it attempts to treat 'dd' as the requested type. Otherwise, it looks in 'dd' for a

value that matches that 'id'. As described above, 'id' has different meanings based on the type of 'dd'.

Return values

DDS_RETCODE_BAD_PARAMETER	in the case of an invalid parameter or non-existent 'id'.

1.9.2.35 DDS_ReturnCode_t DDS_DynamicData_set_char32_value (DDS_DynamicData *dd,* const DDS_MemberId *id,* const cdx_char32_t *value*) [related]

Set a 32bit character value, identified by 'id', in the sample.

If 'id' is DDS_MEMBER_ID_INVALID, then it attempts to treat 'dd' as the requested type. Otherwise, it looks in 'dd' for a value that matches that 'id'. As described above, 'id' has different meanings based on the type of 'dd'.

Return values

DDS_RETCODE_BAD_PARAMETER	in the case of an invalid parameter or non-existent 'id'.

1.9.2.36 DDS_ReturnCode_t DDS_DynamicData_set_char8_value (DDS_DynamicData *dd,* const DDS_MemberId *id,* const char *value*) [related]

Set an 8bit character value, identified by 'id', in the sample.

If 'id' is DDS_MEMBER_ID_INVALID, then it attempts to treat 'dd' as the requested type. Otherwise, it looks in 'dd' for a value that matches that 'id'. As described above, 'id' has different meanings based on the type of 'dd'.

Return values

DDS_RETCODE_BAD_PARAMETER	in the case of an invalid parameter or non-existent 'id'.

1.9.2.37 DDS_ReturnCode_t DDS_DynamicData_set_complex_value (DDS_DynamicData *dd,* const DDS_MemberId *id,* const DDS_DynamicData *value*) [related]

Set a complex value, identified by 'id', in the sample.

If 'id' is DDS_MEMBER_ID_INVALID, then it attempts to treat 'dd' as the requested type. Otherwise, it looks in 'dd' for a value that matches that 'id'. As described above, 'id' has different meanings based on the type of 'dd'.

Return values

DDS_RETCODE_BAD_PARAMETER	in the case of an invalid parameter or non-existent 'id'.

1.9.2.38 DDS_ReturnCode_t DDS_DynamicData_set_float32_value (DDS_DynamicData *dd,* const DDS_MemberId *id,* const float *value*) [related]

Set a float value, identified by 'id', in the sample.

If 'id' is DDS_MEMBER_ID_INVALID, then it attempts to treat 'dd' as the requested type. Otherwise, it looks in 'dd' for a

value that matches that 'id'. As described above, 'id' has different meanings based on the type of 'dd'.

Return values

DDS_RETCODE_BAD_PARAMETER	in the case of an invalid parameter or non-existent 'id'.

1.9.2.39 DDS_ReturnCode_t DDS_DynamicData_set_float64_value (DDS_DynamicData *dd,* const DDS_MemberId *id,* const double *value*) [related]

Set a double value, identified by 'id', in the sample.

If 'id' is DDS_MEMBER_ID_INVALID, then it attempts to treat 'dd' as the requested type. Otherwise, it looks in 'dd' for a value that matches that 'id'. As described above, 'id' has different meanings based on the type of 'dd'.

Return values

DDS_RETCODE_BAD_PARAMETER	in the case of an invalid parameter or non-existent 'id'.

1.9.2.40 DDS_ReturnCode_t DDS_DynamicData_set_int16_value (DDS_DynamicData *dd,* const DDS_MemberId *id,* const short *value*) [related]

Set a 16bit integer value, identified by 'id', in the sample.

If 'id' is DDS_MEMBER_ID_INVALID, then it attempts to treat 'dd' as the requested type. Otherwise, it looks in 'dd' for a value that matches that 'id'. As described above, 'id' has different meanings based on the type of 'dd'.

Return values

DDS_RETCODE_BAD_PARAMETER	in the case of an invalid parameter or non-existent 'id'.

1.9.2.41 DDS_ReturnCode_t DDS_DynamicData_set_int32_value (DDS_DynamicData *dd,* const DDS_MemberId *id,* const int *value*) [related]

Set a 32bit integer value, identified by 'id', in the sample.

If 'id' is DDS_MEMBER_ID_INVALID, then it attempts to treat 'dd' as the requested type. Otherwise, it looks in 'dd' for a value that matches that 'id'. As described above, 'id' has different meanings based on the type of 'dd'.

Return values

DDS_RETCODE_BAD_PARAMETER	in the case of an invalid parameter or non-existent 'id'.

1.9.2.42 DDS_ReturnCode_t DDS_DynamicData_set_int32_values (DDS_DynamicData *dd,* const DDS_MemberId *id,* const DDS_Int32Seq * *value*) [related]

Modify an array of values in the DDS_DynamicData instance.

If 'dd' is an array type, then 'id' must be DDS_MEMBER_ID_INVALID. If 'dd' is an aggretage type, then 'id' must refer

to a member that is of type array. This copies the data into the DynamicData instance 'dd'. Modifications to the value sequence after this call will not have an effect on the contents of 'dd'.

1.9.2.43 DDS_ReturnCode_t DDS_DynamicData_set_int64_value (DDS_DynamicData *dd,* const DDS_MemberId *id,* const int64_t *value*) [related]

Set a 64bit integer value, identified by 'id', in the sample.

If 'id' is DDS_MEMBER_ID_INVALID, then it attempts to treat 'dd' as the requested type. Otherwise, it looks in 'dd' for a value that matches that 'id'. As described above, 'id' has different meanings based on the type of 'dd'.

Return values

DDS_RETCODE_BAD_PARAMETER	in the case of an invalid parameter or non-existent 'id'.

1.9.2.44 DDS_ReturnCode_t DDS_DynamicData_set_string_value (DDS_DynamicData *dd,* const DDS_MemberId *id,* const char * *value*) [related]

Set a string value, identified by 'id', in the sample.

If 'id' is DDS_MEMBER_ID_INVALID, then it attempts to treat 'dd' as the requested type. Otherwise, it looks in 'dd' for a value that matches that 'id'. As described above, 'id' has different meanings based on the type of 'dd'.

Return values

DDS_RETCODE_BAD_PARAMETER	in the case of an invalid parameter or non-existent 'id'.

1.9.2.45 DDS_ReturnCode_t DDS_DynamicData_set_uint16_value (DDS_DynamicData *dd,* const DDS_MemberId *id,* const unsigned short *value*) [related]

Set an unsigned 16bit integer value, identified by 'id', in the sample.

If 'id' is DDS_MEMBER_ID_INVALID, then it attempts to treat 'dd' as the requested type. Otherwise, it looks in 'dd' for a value that matches that 'id'. As described above, 'id' has different meanings based on the type of 'dd'.

Return values

DDS_RETCODE_BAD_PARAMETER	in the case of an invalid parameter or non-existent 'id'.

1.9.2.46 DDS_ReturnCode_t DDS_DynamicData_set_uint32_value (DDS_DynamicData *dd,* const DDS_MemberId *id,* const unsigned int *value*) [related]

Set an unsigned 32bit integer value, identified by 'id', in the sample.

If 'id' is DDS_MEMBER_ID_INVALID, then it attempts to treat 'dd' as the requested type. Otherwise, it looks in 'dd' for a value that matches that 'id'. As described above, 'id' has different meanings based on the type of 'dd'.

Return values

DDS_RETCODE_BAD_PARAMETER	in the case of an invalid parameter or non-existent 'id'.

1.9.2.47 DDS_ReturnCode_t DDS_DynamicData_set_uint64_value (DDS_DynamicData *dd,* const DDS_MemberId *id,* const uint64_t *value*) [related]

Set an unsigned 64bit integer value, identified by 'id', in the sample.

If 'id' is DDS_MEMBER_ID_INVALID, then it attempts to treat 'dd' as the requested type. Otherwise, it looks in 'dd' for a value that matches that 'id'. As described above, 'id' has different meanings based on the type of 'dd'.

Return values

DDS_RETCODE_BAD_PARAMETER	in the case of an invalid parameter or non-existent 'id'.

1.9.2.48 DDS_ReturnCode_t DDS_DynamicData_set_wstring_value (DDS_DynamicData *dd,* const DDS_MemberId *id,* const cdx_char32_t * *value*) [related]

Set a wstring value, identified by 'id', in the sample.

If 'id' is DDS_MEMBER_ID_INVALID, then it attempts to treat 'dd' as the requested type. Otherwise, it looks in 'dd' for a value that matches that 'id'. As described above, 'id' has different meanings based on the type of 'dd'.

Return values

DDS_RETCODE_BAD_PARAMETER	in the case of an invalid parameter or non-existent 'id'.

1.9.2.49 DDS_DynamicData DDS_DynamicDataFactory_create_data (DDS_DynamicDataFactory *ddf,* DDS_DynamicType *dt*) [related]

Create and return a new data sample. All objects returned by this operation should eventually be deleted by calling DDS_DynamicDataFactory_delete_data.

Parameter type - The type of the sample to create.

1.9.2.50 unsigned char DDS_DynamicType_equals (DDS_DynamicType *dt,* const DDS_DynamicType *other*) [related]

Two types shall be considered equal if and only if all of their respective properties are equal.

Return values

zero	if not equal
non-zero	if equal

1.9.2.51 DDS_ReturnCode_t DDS_DynamicType_get_all_members (DDS_DynamicType *dt,* DDS_DynamicTypeMembersById * *member*) [related]

This operation provides access to the 'members_by_id' map.

Return values

DDS_RETCODE_PRECONDITION_NOT_MET	if the type does not have members
DDS_RETCODE_BAD_PARAMETER	in the case of an invalid parameter
DDS_RETCODE_OK	on success

1.9.2.52 DDS_ReturnCode_t DDS_DynamicType_get_all_members_by_name (DDS_DynamicType *dt*, DDS_DynamicTypeMembersByName * *member*) [related]

This operation provides access to the 'members_by_name' map.

Return values

DDS_RETCODE_PRECONDITION_NOT_MET	if the type does not have members
DDS_RETCODE_BAD_PARAMETER	in the case of an invalid parameter
DDS_RETCODE_OK	on success

1.9.2.53 DDS_ReturnCode_t DDS_DynamicType_get_annotation (DDS_DynamicType *dt*, DDS_AnnotationDescriptor * *descriptor*, const unsigned int *idx*) [related]

Access an annotation at the specified index. On success, the 'descriptor' parameter is set to the annotation.

Not Yet Supported This operation is not yet implemented.

1.9.2.54 unsigned int DDS_DynamicType_get_annotation_count (DDS_DynamicType *dt*) [related]

Return the number of annotations applied to this type.

Return values

uint	the number of annotations

Not Yet Supported This operation is not yet implemented.

1.9.2.55 DDS_ReturnCode_t DDS_DynamicType_get_descriptor (DDS_DynamicType *dt*, DDS_TypeDescriptor * *descriptor*) [related]

This operation provides a summary of the current state of this type.

It populates the provided 'descriptor' parameter.

Return values

DDS_RETCODE_BAD_PARAMETER	if any parameters are invalid
DDS_RETCODE_OK	on success

1.9.2.56 DDS_TypeKind DDS_DynamicType_get_kind (DDS_DynamicType *dt*) [related]

This convenience operation returns the kind of this type (e.g., integer, structure, etc.).

Its result shall be the kind indicated by the type's descriptor property.

1.9.2.57 DDS_ReturnCode_t DDS_DynamicType_get_member (DDS_DynamicType *dt,* DDS_DynamicTypeMember * *member,* const DDS_MemberId *id*) [related]

This operation accesses a member by id.

Several types are considered to have 'members"

- structures, unions, enumerations, annotations, and bitsets

On success, the 'member' parameter is set to the member that has a matching member id.

Return values

DDS_RETCODE_PRECONDITION_NOT_MET	if the type does not have members
DDS_RETCODE_BAD_PARAMETER	in the case of an invalid parameter
DDS_RETCODE_ERROR	if the member id is not present in the type
DDS_RETCODE_OK	on success

1.9.2.58 DDS_ReturnCode_t DDS_DynamicType_get_member_by_name (DDS_DynamicType *dt,* DDS_DynamicTypeMember * *member,* const DDS_ObjectName *name*) [related]

This operation accesses a member by name.

Several types are considered to have 'members"

- structures, unions, enumerations, annotations, and bitsets

On success, the 'member' parameter is set to the named member.

Return values

DDS_RETCODE_PRECONDITION_NOT_MET	if the type does not have members
DDS_RETCODE_BAD_PARAMETER	in the case of an invalid parameter
DDS_RETCODE_ERROR	if the named member is not present in the type
DDS_RETCODE_OK	on success

1.9.2.59 const char * DDS_DynamicType_get_name (DDS_DynamicType *dt*) [related]

This convenience operation provides the fully qualified name of this type.

It is identical to the name string that is a member of the descriptor property.

1.9.2.60 DDS_ReturnCode_t DDS_DynamicTypeBuilder_add_member (DDS_DynamicTypeBuilder *dtb,* const DDS_MemberDescriptor * *descriptor*) [related]

Add a member to the type.

Return values

DDS_RETCODE_PRECONDITION_NOT_MET	if the type does not have members or a member with the same id already exists
DDS_RETCODE_BAD_PARAMETER	in the case of an invalid parameter
DDS_RETCODE_OUT_OF_RESOURCES	in the case of a memory shortage
DDS_RETCODE_OK	on success

1.9.2.61 DDS_ReturnCode_t DDS_DynamicTypeBuilder_apply_annotation (DDS_DynamicTypeBuilder *dtb,* const DDS_AnnotationDescriptor * *descriptor*) [related]

Apply the given annotation to this type.

Parameter descriptor - A consistent descriptor for the annotation to apply.

Return values

DDS_RETCODE_BAD_PARAMETER	If this 'descriptor' parameter is not consistent

Not Yet Supported This operation is not yet implemented.

1.9.2.62 DDS_DynamicType DDS_DynamicTypeBuilder_build (DDS_DynamicTypeBuilder *dtb*) [related]

Create an immutable DDS_DynamicType object defined by the builder's current state.

Subsequent changes to this builder, if any, shall have no effect on the state of any previously created DDS_DynamicType.

Return values

nil	in the case of an error

1.9.2.63 DDS_ReturnCode_t DDS_DynamicTypeBuilder_delete_type (DDS_DynamicTypeBuilder *dtb,* DDS_DynamicType *type*) [related]

Destroy the type information contained in this DDS_DynamicTypeBuilder instance.

Return values

DDS_RETCODE_BAD_PARAMETER	in the case of an invalid parameter
DDS_RETCODE_OK	on success

1.9.2.64 unsigned char DDS_DynamicTypeBuilder_equals (DDS_DynamicTypeBuilder *dtb,* const DDS_DynamicType *other*) [related]

Two types shall be considered equal if and only if all of their respective properties are equal.

Return values

zero	if not equal
non-zero	if equal

1.9.2.65 DDS_ReturnCode_t DDS_DynamicTypeBuilder_get_all_members (DDS_DynamicTypeBuilder *dtb,* DDS_DynamicTypeMembersById * *member*) [related]

This operation provides access to the 'members_by_id' map.

Return values

DDS_RETCODE_PRECONDITION_NOT_MET	if the type does not have members
DDS_RETCODE_BAD_PARAMETER	in the case of an invalid parameter
DDS_RETCODE_OK	on success

1.9.2.66 DDS_ReturnCode_t DDS_DynamicTypeBuilder_get_all_members_by_name (DDS_DynamicTypeBuilder *dtb,* DDS_DynamicTypeMembersByName * *member*) [related]

This operation provides access to the 'members_by_name' map.

Return values

DDS_RETCODE_PRECONDITION_NOT_MET	if the type does not have members
DDS_RETCODE_BAD_PARAMETER	in the case of an invalid parameter
DDS_RETCODE_OK	on success

1.9.2.67 DDS_ReturnCode_t DDS_DynamicTypeBuilder_get_annotation (DDS_DynamicTypeBuilder *dtb,* DDS_AnnotationDescriptor * *descriptor,* const unsigned int *idx*) [related]

Access an annotation at the specified index. On success, the 'descriptor' parameter is set to the annotation.

Not Yet Supported This operation is not yet implemented.

1.9.2.68 unsigned int DDS_DynamicTypeBuilder_get_annotation_count (DDS_DynamicTypeBuilder *dtb*) [related]

Return the number of annotations applied to this type.

Return values

uint	the number of annotations

Not Yet Supported This operation is not yet implemented.

1.9.2.69 DDS_ReturnCode_t DDS_DynamicTypeBuilder_get_descriptor (DDS_DynamicTypeBuilder *dtb,* DDS_TypeDescriptor * *descriptor*) [related]

This operation provides a summary of the state of this type.

It overwrites the state of the application-provided 'descriptor' object.

Return values

DDS_RETCODE_BAD_PARAMETER	if the descriptor parameter is invalid
DDS_RETCODE_OK	on success

1.9.2.70 DDS_TypeKind DDS_DynamicTypeBuilder_get_kind (DDS_DynamicTypeBuilder *dtb*) [related]

This convenience operation indicates the kind of this type (e.g., integer, structure, etc.).

Its result shall be the same as the kind indicated by the type's descriptor property.

1.9.2.71 DDS_ReturnCode_t DDS_DynamicTypeBuilder_get_member (DDS_DynamicTypeBuilder *dtb,* DDS_DynamicTypeMember * *member,* const DDS_MemberId *id*) [related]

This operation accesses a member by id.

Several types are considered to have 'members"

- structures, unions, enumerations, annotations, and bitsets

On success, the 'member' parameter is set to the member that has a matching member id.

Return values

DDS_RETCODE_PRECONDITION_NOT_MET	if the type does not have members
DDS_RETCODE_BAD_PARAMETER	in the case of an invalid parameter
DDS_RETCODE_ERROR	if the member id is not present in the type
DDS_RETCODE_OK	on success

1.9.2.72 DDS_ReturnCode_t DDS_DynamicTypeBuilder_get_member_by_name (DDS_DynamicTypeBuilder *dtb,* DDS_DynamicTypeMember * *member,* const DDS_ObjectName *name*) [related]

This operation accesses a member by name.

Several types are considered to have 'members"

- structures, unions, enumerations, annotations, and bitsets

On success, the 'member' parameter is set to the named member.

Return values

DDS_RETCODE_PRECONDITION_NOT_MET	if the type does not have members
DDS_RETCODE_BAD_PARAMETER	in the case of an invalid parameter
DDS_RETCODE_ERROR	if the named member is not present in the type
DDS_RETCODE_OK	on success

1.9.2.73 const char * DDS_DynamicTypeBuilder_get_name (DDS_DynamicTypeBuilder *dtb*) [related]

This convenience operation provides the fully qualified name of this type. I.

t shall be identical to the name string that is a member of the descriptor property.

1.9.2.74 DDS_ReturnCode_t DDS_DynamicTypeBuilder_set_extensibility (DDS_DynamicTypeBuilder *dtb,* uint32_t *ext*) [related]

Assign the specified extensibility to the Builder (struct/union)

The 'ext' parameter is 0, 1, or 2, for FINAL, EXTENSIBLE, and MUTABLE, respectively.

Return values

DDS_RETCODE_BAD_PARAMETER	If the 'ext' parameter is invalid.

1.9.2.75 DDS_DynamicTypeBuilder DDS_DynamicTypeBuilderFactory_create_alias_type (DDS_DynamicTypeBuilderFactory *dtbf,* const DDS_DynamicType *base_type*) [related]

Create and return a new DDS_DynamicTypeBuilder object representing an alias for the provided 'base_type'.

This is a convenience routine, in place of calling DDS_DynamicTypeBuilder_create_type() directly. The alias type refers to the 'base_type' parameter.

Return values

nil	In the case of an error

1.9.2.76 DDS_DynamicTypeBuilder DDS_DynamicTypeBuilderFactory_create_array_type (DDS_DynamicTypeBuilderFactory *dtbf,* const DDS_DynamicType *element_type,* const DDS_BoundSeq * *bound*) [related]

Create and return a new DDS_DynamicTypeBuilder object representing an array type.

All objects returned by this operation should eventually be deleted by calling DDS_DynamicTypeBuilderFactory_delete_type_builder.

All array types having equal element types, an equal number of dimensions, and equal bounds in each dimension shall be considered equal. An implementation may therefore elect whether to always return a new object from this method or whether to pool objects and to return previously created type objects consistent with these rules.

The type of all objects that can be stored in an array of the new type is specified by parameter 'element_type'.

The 'bound' parameter is a collection of unsigned integers, the length of which is equal to the number of dimensions in the new array type, and the values of which are the bounds of each dimension. (For example, a three-by-two array would be described by a collection of length two, where the first element had a value of three and the second a value of two.)

Return values

nil	If any parameters are invalid or an error occurs.

1.9.2.77 DDS_DynamicTypeBuilder DDS_DynamicTypeBuilderFactory_create_bitset_type (DDS_DynamicTypeBuilderFactory *dtbf,* const unsigned int *bound*) [related]

Create and return a new DDS_DynamicTypeBuilder object representing a bit set type.

All objects returned by this operation should eventually be deleted by calling DDS_DynamicTypeBuilderFactory_delete_type_builder.

Parameter bound - The number of reserved bits in the bit set.

Return values

nil	If an error occurs, for example if the bound is out of range.

1.9.2.78 DDS_DynamicTypeBuilder DDS_DynamicTypeBuilderFactory_create_enumeration_type (DDS_DynamicTypeBuilderFactory *dtbf,* const unsigned int *bound*) [related]

Create and return a new DDS_DynamicTypeBuilder object representing an enumeration type.

This is a convenience routine, in place of calling DDS_DynamicTypeBuilder_create_type() directly. The enumeration type of the result returned shall initially contain no members.

Return values

nil	In the case of an error

1.9.2.79 DDS_ReturnCode_t DDS_DynamicTypeBuilderFactory_create_extensibility_annotation (DDS_DynamicTypeBuilderFactory *dtbf,* DDS_AnnotationDescriptor * *ad,* DDS_ExtensibilityKind *ekind*) [related]

Initialize an AnnotationDescriptor to represent an '@extensibility' annotation with the specified extensibility kind.

This is a convenience routine, in place of calling DDS_DynamicTypeBuilderFactory routines directly. Use DDS_AnnotationDescriptor_clear() to reclaim the memory allocated by this routine.

Return values

DDS_RETCODE_OK	if AnnotationDescriptor is correctly initialized
DDS_RETCODE_BAD_PARAMETER	in the case of an invalid parameter (ie, dtbf or ad == NULL)

1.9.2.80 DDS_DynamicTypeBuilder DDS_DynamicTypeBuilderFactory_create_map_type (DDS_DynamicTypeBuilderFactory *dtbf,* const DDS_DynamicType *key_element_type,* const DDS_DynamicType *element_type,* const unsigned int *bound*) [related]

Create and return a new DDS_DynamicTypeBuilder object representing a map type.

All objects returned by this operation should eventually be deleted by calling DDS_DynamicTypeBuilderFactory_delete_type_builder.

All map types having equal key and value element types and equal bounds shall be considered equal. An implementation may therefore elect whether to always return a new object from this method or whether to pool objects and to return previously created type objects consistent with these rules.

Parameter key_element_type - The type of all objects that can be stored as keys in a map of the new type.

Parameter element_type - The type of all objects that can be stored as values in a map of the new type.

Parameter bound - The maximum number of key-value pairs that may be stored in a map of the new type. If this argument is equal to LENGTH_UNLIMITED, the map type shall be considered to be unbounded.

Return values

nil	If an error occurs, or any parameter is invalid

1.9.2.81 DDS_ReturnCode_t DDS_DynamicTypeBuilderFactory_create_optional_annotation (DDS_DynamicTypeBuilderFactory *dtbf,* DDS_AnnotationDescriptor * *ad*) [related]

Initialize an AnnotationDescriptor to represent an '@optional' annotation.

This is a convenience routine, in place of calling DDS_DynamicTypeBuilderFactory routines directly. Use DDS_AnnotationDescriptor_clear() to reclaim the memory allocated by this routine.

Return values

DDS_RETCODE_OK	if AnnotationDescriptor is correctly initialized
DDS_RETCODE_BAD_PARAMETER	in the case of an invalid parameter (ie, dtbf or ad == NULL)

1.9.2.82 DDS_DynamicTypeBuilder DDS_DynamicTypeBuilderFactory_create_sequence_type (DDS_DynamicTypeBuilderFactory *dtbf,* const DDS_DynamicType *element_type,* const unsigned int *bound*) [related]

Create and return a new DDS_DynamicTypeBuilder object representing a sequence type.

All objects returned by this operation should eventually be deleted by calling DDS_DynamicTypeBuilderFactory_delete_type_builder.

All sequence types having equal element types and equal bounds shall be considered equal. An implementation may therefore elect whether to always return a new object from this method or whether to pool objects and to return previously created type objects consistent with these rules.

The type of all objects that can be stored in a sequence of the new type is specified by the 'element_type' parameter.

The maximum number of elements that may be stored in a sequence of the new type is specified by the 'bound' parameter. If this argument is equal to DDS_LENGTH_UNLIMITED, the sequence type shall be considered to be unbounded.

Return values

nil	If an error occurs, or any parameters are invalid

1.9.2.83 DDS_DynamicTypeBuilder DDS_DynamicTypeBuilderFactory_create_string_type (DDS_DynamicTypeBuilderFactory *dtbf,* const unsigned int *bound*) [related]

Create and return a new DDS_DynamicTypeBuilder object representing a string type.

The element type of the result returned by DDS_DynamicTypeBuilderFactory_create_string_type shall be Char8. The element type of the result returned by create_wstring_type shall be Char32.

All string types having equal element types and equal bounds shall be considered equal. An implementation may therefore elect whether to always return a new object from this method or whether to pool objects and to return previously created type objects consistent with these rules.

The maximum number of elements that may be stored in a string of the new type is specified by parameter 'bound'. If this argument is equal to DDS_LENGTH_UNLIMITED, the string type shall be unbounded.

Return values

nil	If an error occurs

1.9.2.84 DDS_DynamicTypeBuilder DDS_DynamicTypeBuilderFactory_create_structure_type (DDS_DynamicTypeBuilderFactory *dtbf,* DDS_DynamicType *base_type*) [related]

Create and return a new DDS_DynamicTypeBuilder object representing a structure type.

This is a convenience routine, in place of calling DDS_DynamicTypeBuilder_create_type() directly. The structure type of the result returned shall initially contain no members. If the structure is to inherit from a parent type, it is specified in the 'base_type' parameter, otherwise, 'base_type' should be set tp NULL.

Return values

nil	In the case of an error

1.9.2.85 DDS_DynamicTypeBuilder DDS_DynamicTypeBuilderFactory_create_type (DDS_DynamicTypeBuilderFactory *dtbf,* const DDS_TypeDescriptor * *descriptor*) [related]

Create and return a new DDS_DynamicTypeBuilder object as described by the given type descriptor.

This method is the conventional mechanism for creating structured, enumeration, and alias types, although it can also be used to create types of other kinds. All objects returned by this operation should eventually be deleted by calling DDS_DynamicTypeBuilderFactory_delete_type.

The properties of the new type to create is provided by the 'descriptor' parameter.

Return values

nil	If the descriptor parameter is nil or inconsistent (as indicated by its is_consistent operation)

1.9.2.86 DDS_DynamicTypeBuilder DDS_DynamicTypeBuilderFactory_create_type_copy (DDS_DynamicTypeBuilderFactory *dtbf,* const DDS_DynamicType *type*) [related]

Create and return a new DDS_DynamicTypeBuilder object with a copy of the state of the given type.

All objects returned by this operation should eventually be deleted by calling DDS_DynamicTypeBuilderFactory_delete_type_builder. The initial state of the new type to create is described by the parameter 'type'.

Return values

nil	If parameter 'type' is nil

1.9.2.87 DDS_DynamicTypeBuilder DDS_DynamicTypeBuilderFactory_create_type_w_document (DDS_DynamicTypeBuilderFactory *dtbf,* const char * *document,* const char * *type_name,* const DDS_IncludePathSeq * *include_paths*) [related]

Create and return a new DDS_DynamicTypeBuilder object by parsing the type description contained in the given string.

Applications shall be able to reclaim resources associated with the type returned by this method by calling DDS_DynamicTypeBuilderFactory_delete_type_builder, just as if the resultant type was created by one of the create methods of this class.

Parameter document - A type description document, which shall be parsed to create the DDS_DynamicType object. Implementations shall minimally support the XML Type Description format for loaded documents and may support additional Type Descriptions. The 'document' parameter shall be a NUL terminated string containing the XML text of the document.

Parameter type_name - The fully qualified name of the type to be loaded from the document. If no type exists of this name in the document, the operation shall fail and return a nil result.

Parameter include_paths - A collection of URLs to directories to be searched for additional type description documents that may be included, directly or indirectly, by the document argument. Implementations shall minimally support the file: URL scheme and may support additional schemes.

Return values

nil	If an error occurs

Not Yet Supported This operation is not yet implemented.

1.9.2.88 DDS_DynamicTypeBuilder DDS_DynamicTypeBuilderFactory_create_type_w_type_object (DDS_DynamicTypeBuilderFactory *dtbf,* const DDS_TypeObject * *type_object*) [related]

Create and return a new DDS_DynamicTypeBuilder object that describes a type identical to that described by the given TypeObject.

Subsequent changes to the new DDS_DynamicTypeBuilder object shall not impact the input TypeObject 'type_object'.
All objects returned by this operation should eventually be deleted by calling DDS_DynamicTypeBuilderFactory_delete_type_builder.

Return values

nil	If the type_object parameter is invalid

1.9.2.89 DDS_DynamicTypeBuilder DDS_DynamicTypeBuilderFactory_create_type_w_uri (DDS_DynamicTypeBuilderFactory *dtbf,* const char * *document_url,* const char * *type_name,* const DDS_IncludePathSeq * *include_paths*) [related]

Create and return a new DDS_DynamicTypeBuilder object by parsing the type description at the given URL.

Applications shall be able to reclaim resources associated with the type returned by this method by calling DDS_DynamicTypeBuilderFactory_delete_type_builder, just as if the resultant type was created by one of the create methods of this class.

Parameter document_uri - A URL that indicates a type description document, which shall be parsed to create the DDS_DynamicType object. Implementations shall minimally support the "file:" URL scheme and may support additional schemes. Implementations shall minimally support the XML Type Description format for loaded documents and may support additional Type Descriptions.

Parameter type_name - The fully qualified name of the type to be loaded from the document that is the target of the URL. If no type exists of this name in the document, the operation shall fail and return a nil result.

Parameter include_paths - A collection of URLs to directories to be searched for additional type description documents that may be included, directly or indirectly, by the document that is the target of document_url. The directory in which the target of document_url resides shall be considered on the inclusion search path implicitly and need not be included in this collection. Implementations shall minimally support the file: URL scheme and may support additional schemes.

Return values

nil	If an error occurs, or if any parameters are invalid

Not Yet Supported This operation is not yet implemented.

1.9.2.90 DDS_DynamicTypeBuilder DDS_DynamicTypeBuilderFactory_create_union_type (DDS_DynamicTypeBuilderFactory *dtbf*) [related]

Create and return a new DDS_DynamicTypeBuilder object representing a union type.

This is a convenience routine, in place of calling DDS_DynamicTypeBuilder_create_type() directly. The union type of the result returned shall initially contain no members.

Return values

nil	In the case of an error

1.9.2.91 DDS_DynamicTypeBuilder DDS_DynamicTypeBuilderFactory_create_wstring_type (DDS_DynamicTypeBuilderFactory *dtbf,* const unsigned int *bound*) [related]

Create and return a new DDS_DynamicTypeBuilder object representing a string type.

The element type of the result returned by DDS_DynamicTypeBuilderFactory_create_string_type shall be Char8. The element type of the result returned by create_wstring_type shall be Char32.

All string types having equal element types and equal bounds shall be considered equal. An implementation may therefore elect whether to always return a new object from this method or whether to pool objects and to return previously created type objects consistent with these rules.

The maximum number of elements that may be stored in a string of the new type is specified by parameter 'bound'. If this argument is equal to DDS_LENGTH_UNLIMITED, the string type shall be unbounded.

Return values

nil	If an error occurs

1.9.2.92 DDS_ReturnCode_t DDS_DynamicTypeBuilderFactory_delete_instance (void) [related]

Reclaim any resources associated with the instance returned from DDS_DynamicTypeBuilderFactory_get_instance.

Return values

RETCODE_OK	on success
RETCODE_ERROR	if it fails

1.9.2.93 DDS_ReturnCode_t DDS_DynamicTypeBuilderFactory_delete_type (DDS_DynamicTypeBuilderFactory *dtbf,* DDS_DynamicType *type*) [related]

Destroy a DDS_DynamicType instance obtained through the DDS_DynamicTypeBuilderFactory_get_primitive_type operation.

Return values

DDS_RETCODE_OK	on success
DDS_RETCODE_ALREADY_DELETED	if the type has previously been deleted and the middlware can detect that fact
DDS_RETCODE_ERROR	on any other error

1.9.2.94 DDS_ReturnCode_t DDS_DynamicTypeBuilderFactory_delete_type_builder (DDS_DynamicTypeBuilderFactory *dtbf,* DDS_DynamicTypeBuilder *type_builder*) [related]

Destroy a DDS_DynamicTypeBuilder instance obtained through one of the create methods on the DDS_DynamicTypeBuilderFactory.

Return values

DDS_RETCODE_OK	on success
DDS_RETCODE_ERROR	on any other error

1.9.2.95 DDS_DynamicTypeBuilderFactory DDS_DynamicTypeBuilderFactory_get_instance (void) [related]

Return the DDS_DynamicTypeBuilderFactory singleton instance.

Calling this operation is legal even after delete_instance has been called. In such a case, the implementation shall recreate or restore the state of the "singleton" as necessary in order to return a valid object to the caller. If an error occurs, this method shall return a nil value.

1.9.2.96 DDS_DynamicType DDS_DynamicTypeBuilderFactory_get_primitive_type (DDS_DynamicTypeBuilderFactory *dtbf,* const DDS_TypeKind *kind*) [related]

Retrieve a DDS_DynamicType object corresponding to the indicated primitive type kind.

The kind of the primitive type whose representation is to be returned is specified by parameter 'kind'.

Return values

nil	if the given kind does not correspond to a primitive type

1.9.2.97 unsigned char DDS_DynamicTypeMember_equals (DDS_DynamicTypeMember *dtm,* const DDS_DynamicTypeMember *other*) [related]

Compare two DDS_DynamicTypeMember's.

Two type descriptors are considered equal if and only if if they belong to the same type and the values of all of the properties are equal in each of them.

1.9.2.98 DDS_ReturnCode_t DDS_DynamicTypeMember_get_annotation (DDS_DynamicTypeMember *dtm,* DDS_AnnotationDescriptor * *descriptor,* const unsigned int *idx*) [related]

Access an annotation at the specified index. On success, the 'descriptor' parameter is set to the annotation.

Not Yet Supported This operation is not yet implemented.

1.9.2.99 unsigned int DDS_DynamicTypeMember_get_annotation_count (DDS_DynamicTypeMember *dtm*) [related]

Return the number of annotations applied to this type.

Return values

uint	the number of annotations

Not Yet Supported This operation is not yet implemented.

1.9.2.100 DDS_ReturnCode_t DDS_DynamicTypeMember_get_descriptor (DDS_DynamicTypeMember *dtm,* DDS_MemberDescriptor * *descriptor*) [related]

This operation accesses the current state of this type.

It overwrites the application-provided 'descriptor' parameter.

Return values

DDS_RETCODE_BAD_PARAMETER	in the case of an invalid parameter

1.9.2.101 DDS_MemberId DDS_DynamicTypeMember_get_id (DDS_DynamicTypeMember *dtm*) [related]

This convenience operation provides the member ID of this member.

Its result shall be identical to the ID value that is a member of the descriptor property.

1.9.2.102 const char * DDS_DynamicTypeMember_get_name (DDS_DynamicTypeMember *dtm*) [related]

This convenience operation provides the name of this member.

Its result shall be identical to the name string that is a member of the descriptor property.

1.9.2.103 DDS_DynamicTypeSupport DDS_DynamicTypeSupport_create_type_support (const DDS_DynamicType *type*) [related]

Create and return a new DDS_DynamicTypeSupport object capable of supporting the given type.

The new type support has a "copy" of the given 'type' parameter, such that subsequent changes to, or deletions of, the 'type' parameter do not impact the new type support.

Any object returned by this operation should eventually be deleted by calling DDS_DynamicTypeSupport_delete_type_support.

Parameter type - The type for which to create a type support.

Return values

nil	If an error occurs, or the parameter is invalid

1.9.2.104 DDS_ReturnCode_t DDS_DynamicTypeSupport_delete_type_support (DDS_DynamicTypeSupport *type_support*) [related]

Delete the given type support object, which was previously created by a call to DDS_DynamicTypeSupport_create_type_support.

Return values

DDS_RETCODE_BAD_PARAMETER	invalid parameter
DDS_RETCODE_OK	on success

1.9.2.105 DDS_DynamicType DDS_DynamicTypeSupport_get_type (DDS_DynamicTypeSupport *dts*) [related]

Returns the DDS_DynamicType that is supported by this TypeSupport instance.

The application should not modify the returned DDS_DynamicType.

1.9.2.106 const char * DDS_DynamicTypeSupport_get_type_name (DDS_DynamicTypeSupport *dts*) [related]

Returns the name of the type supported by this TypeSupport instance.

The returned string is owned by the TypeSupport, and must not be free'd or modified.

1.9.2.107 DDS_ReturnCode_t DDS_MemberDescriptor_copy_from (DDS_MemberDescriptor * *to,* const DDS_MemberDescriptor * *from*) [related]

Overwrite the contents of the 'to' descriptor with those of the provided 'from' descriptor.

Subsequent calls to equals, passing the same argument as to this method, return true.

The 'from' descriptor shall not be changed by this operation.

Return values

DDS_RETCODE_BAD_PARAMETER	in the case of an invalid parameter
DDS_RETCODE_ERROR	in the case of any other error

1.9.2.108 unsigned char DDS_MemberDescriptor_equals (DDS_MemberDescriptor * *md,* const DDS_MemberDescriptor * *other*) [related]

Compare two DDS_TypeDescriptor's.

Two type descriptors are considered equal if and only if the values of all of the properties identified in the table above are equal in each of them.

1.9.2.109 unsigned char DDS_MemberDescriptor_is_consistent (DDS_MemberDescriptor * *md*) [related]

Indicates whether the states of all of this descriptor's properties are consistent.

The definitions of consistency for each property are given in the section corresponding to that property.

Return values

non-zero	if descriptor is consistent
zero	if descriptor is not consistent

1.9.2.110 DDS_ReturnCode_t DDS_TypeDescriptor_copy_from (DDS_TypeDescriptor * *to,* const DDS_TypeDescriptor * *from*) [related]

Overwrite the contents of this descriptor with those of another descriptor.

Subsequent calls to equals, passing the same argument as to this method, return true. The other descriptor shall not be changed by this operation.

Return values

DDS_RETCODE_BAD_PARAMETER	in the case of an invalid parameter
DDS_RETCODE_ERROR	in the case of any other error

1.9.2.111 unsigned char DDS_TypeDescriptor_equals (const DDS_TypeDescriptor * *td*, const DDS_TypeDescriptor * *other*) [related]

Compare two DDS_TypeDescriptor's.

Two type descriptors are considered equal if and only if the values of all of the properties identified in the table above are equal in each of them.

1.9.2.112 unsigned char DDS_TypeDescriptor_is_consistent (const DDS_TypeDescriptor * *td*) [related]

Indicates whether the states of all of this descriptor's properties are consistent.

The definitions of consistency for each property are given in the section corresponding to that property.

Return values

non-zero	if descriptor is consistent
zero	if descriptor is not consistent

1.10 CoreDX DDS DynamicTypes

Data Structures

- struct CDX_DynamicType

  CDX_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).
- struct CDX_DynamicTypeDataReader

  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.
- struct CDX_DynamicTypeDataWriter

  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.

Enumerations

1.10.1 Detailed Description

See also

CDX_DynamicTypeTypeSupport

1.10.2 Enumeration Type Documentation

1.10.2.1 enum DDS_TypeCodeKind

Identifies the different data types supported by DDS_DynamicType objects.

Enumerator

DDS_TYPECODE_SHORT Typecode to identify a 'short' data type (16 bits with sign)

DDS_TYPECODE_LONG Typecode to identify a 'long' data type (32 bits with sign)

DDS_TYPECODE_USHORT Typecode to identify an 'unsigned short' data type (16 bits without sign)

DDS_TYPECODE_ULONG Typecode to identify an 'unsigned long' data type (32 bits without sign)

DDS_TYPECODE_FLOAT Typecode to identify a 'float' data type

DDS_TYPECODE_DOUBLE Typecode to identify a 'double' data type

DDS_TYPECODE_BOOLEAN Typecode to identify a 'boolean' data type

DDS_TYPECODE_CHAR Typecode to identify a 'char' data type (8 bits)

DDS_TYPECODE_OCTET Typecode to identify an 'octet' data type (8 bits)

DDS_TYPECODE_STRUCT Typecode to identify a 'struct' data type (with named sub-fields)

DDS_TYPECODE_UNION Typecode to identify a 'union' data type (with named sub-fields, selected by a discriminator)

DDS_TYPECODE_ENUM Typecode to identify an 'enum' data type

DDS_TYPECODE_STRING Typecode to identify a 'string' data type. May be bounded length or unlimited.

DDS_TYPECODE_SEQUENCE Typecode to identify a 'sequence' data type. May be bounded length or unlimited.

DDS_TYPECODE_ARRAY Typecode to identify an 'array' data type.

DDS_TYPECODE_ALIAS Not used. Included for compatibility.

DDS_TYPECODE_LONGLONG Typecode to identify a 'long long' data type (64 bits with sign)

DDS_TYPECODE_ULONGLONG Typecode to identify an 'unsigned long long' data type (64 bits without sign)

DDS_TYPECODE_LONGDOUBLE Typecode to identify a 'long double' data type. Not supported by CoreDX DDS.

DDS_TYPECODE_WCHAR Typecode to identify a 'wide char' data type.

DDS_TYPECODE_WSTRING Typecode to identify a 'wide string' data type.

DDS_TYPECODE_VALUE Not used. Included for compatibility.

DDS_TYPECODE_VALUE_PARAM Not used. Included for compatibility.

DDS_TYPECODE_ANY Not used. Included for compatibility.

DDS_TYPECODE_FIXEDSTR Alternate to STRING

DDS_TYPECODE_SCOPE Not used. Included for compatibility.

DDS_TYPECODE_SYMBOL Not used. Included for compatibility.

DDS_TYPECODE_MAP Typecode to identify a 'Map' data type. (New from X-Types)

DDS_TYPECODE_BITSET Typecode to identify a 'BitSet' data type. (New from X-Types)

DDS_TYPECODE_ENUM2 Typecode to identify a 2 byte 'enum' data type. (New from X-Types)

1.11 CoreDX DDS DynamicTypeTypeSupport

See also

CDX_DynamicType

Chapter 2

Primary DDS Entities and Types

2.1 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**.*

2.1.1 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.

2.1.2 Friends And Related Function Documentation

2.1.2.1 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.

2.2 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.

2.2.1 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()

2.2.2 Friends And Related Function Documentation

2.2.2.1 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.

2.2.2.2 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.

2.2.2.3 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.

2.3 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_guid (DDS_DataReader dr, DDS_GUID_t *guid)

  Access the GUID which uniquely identifies this reader.
- DDS_ReturnCode_t DDS_DataReader_get_cache_stats (DDS_DataReader dr, DDS_CacheStatistics *stats)

  Provides access to the current cache statistics 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, int32_t 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, int32_t 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, int32_t 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, int32_t 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, int32_t 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, int32_t 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, int32_t 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, int32_t 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, int32_t 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, int32_t 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.

2.3.1 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).

2.3.2 Friends And Related Function Documentation

2.3.2.1 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()

2.3.2.2 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().

2.3.2.3 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.

2.3.2.4 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.

2.3.2.5 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].

2.3.2.6 DDS_ReturnCode_t DDS_DataReader_get_cache_stats (DDS_DataReader *dr,* DDS_CacheStatistics * *stats*) [related]

Provides access to the current cache statistics of the DataReader.

This routine will populate 'stats' with the current statistics.

2.3.2.7 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.

2.3.2.8 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.

2.3.2.9 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.

2.3.2.10 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.

2.3.2.11 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. To reclaim any memory returned in the 'publication_data' parameter, call DDS_DCPSPublication_clear(publication_data).

2.3.2.12 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.

2.3.2.13 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)**.

2.3.2.14 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.

2.3.2.15 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.

2.3.2.16 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.

2.3.2.17 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.

2.3.2.18 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**.

2.3.2.19 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.

2.3.2.20 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.

2.3.2.21 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.

2.3.2.22 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.

2.3.2.23 DDS_ReturnCode_t DDS_DataReader_read (DDS_DataReader *dr,* DDS_PointerSeq * *received_data,* DDS_SampleInfoSeq * *sample_infos,* int32_t *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.

2.3.2.24 DDS_ReturnCode_t DDS_DataReader_read_instance (DDS_DataReader *dr,* DDS_PointerSeq * *received_data,* DDS_SampleInfoSeq * *sample_infos,* int32_t *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()

2.3.2.25 DDS_ReturnCode_t DDS_DataReader_read_next_instance (DDS_DataReader *dr,* DDS_PointerSeq * *received_data,* DDS_SampleInfoSeq * *sample_infos,* int32_t *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()

2.3.2.26 DDS_ReturnCode_t DDS_DataReader_read_next_instance_w_condition (DDS_DataReader *dr,* DDS_PointerSeq * *received_data,* DDS_SampleInfoSeq * *sample_infos,* int32_t *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()

2.3.2.27 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()

2.3.2.28 DDS_ReturnCode_t DDS_DataReader_read_w_condition (DDS_DataReader *dr*, DDS_PointerSeq * *received_data*, DDS_SampleInfoSeq * *sample_infos*, int32_t *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()

2.3.2.29 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.

2.3.2.30 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.

2.3.2.31 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.

2.3.2.32 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.

2.3.2.33 DDS_ReturnCode_t DDS_DataReader_take (DDS_DataReader *dr,* DDS_PointerSeq * *received_data,* DDS_SampleInfoSeq * *sample_infos,* int32_t *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.

2.3.2.34 DDS_ReturnCode_t DDS_DataReader_take_instance (DDS_DataReader *dr,* DDS_PointerSeq * *received_data,* DDS_SampleInfoSeq * *sample_infos,* int32_t *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()

2.3.2.35 DDS_ReturnCode_t DDS_DataReader_take_next_instance (DDS_DataReader *dr,* DDS_PointerSeq * *received_data,* DDS_SampleInfoSeq * *sample_infos,* int32_t *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()

2.3.2.36 DDS_ReturnCode_t DDS_DataReader_take_next_instance_w_condition (DDS_DataReader *dr,* DDS_PointerSeq * *received_data,* DDS_SampleInfoSeq * *sample_infos,* int32_t *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()

2.3.2.37 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()

2.3.2.38 DDS_ReturnCode_t DDS_DataReader_take_w_condition (DDS_DataReader *dr,* DDS_PointerSeq * *received_data,* DDS_SampleInfoSeq * *sample_infos,* int32_t *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()

2.4 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_ReturnCode_t DDS_DataWriter_wait_for_acknowledgments (DDS_DataWriter dw, const DDS_Duration_t *max_wait)

  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_OfferedDeadlineMissedSta ∗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_OfferedIncompatibleQosS ∗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_cache_stats (DDS_DataWriter dw, DDS_CacheStatistics ∗stats)

  Provides access to the current cache statistics of the DataWriter.
- DDS_ReturnCode_t DDS_DataWriter_get_guid (DDS_DataWriter dw, DDS_GUID_t ∗guid)

  Access the GUID which uniquely identifies this writer.
- 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.

2.4.1 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.

2.4.2 Friends And Related Function Documentation

2.4.2.1 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.

2.4.2.2 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.

2.4.2.3 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.

2.4.2.4 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].

2.4.2.5 DDS_ReturnCode_t DDS_DataWriter_get_cache_stats (DDS_DataWriter *dw,* DDS_CacheStatistics * *stats*) [related]

Provides access to the current cache statistics of the DataWriter.

This routine will populate 'stats' with the current statistics.

2.4.2.6 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.

2.4.2.7 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.

2.4.2.8 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.

2.4.2.9 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.

2.4.2.10 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. To reclaim any memory returned in the 'subscription_data' parameter, call DDS_DCPSSubscription_clear(subscription_data).

2.4.2.11 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.

2.4.2.12 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.

2.4.2.13 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.

2.4.2.14 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.

2.4.2.15 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)**.

2.4.2.16 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**.

2.4.2.17 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.

2.4.2.18 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.

2.4.2.19 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.

2.4.2.20 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.

2.4.2.21 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.

2.4.2.22 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.

2.4.2.23 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.

2.4.2.24 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.

2.4.2.25 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.

2.4.2.26 DDS_ReturnCode_t DDS_DataWriter_wait_for_acknowledgments (DDS_DataWriter *dw,* const DDS_Duration_t * *max_wait*) [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'

2.4.2.27 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.

2.4.2.28 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.

2.5 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_ex (DDS_DomainParticipantFactory dpf, DDS_DomainId_t domain_id, DDS_DomainParticipantQos *qos, DDS_DomainParticipantListener *a_listener, DDS_StatusMask mask)

  This operation creates a new DomainParticipant object.
- 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_set_license_debug (unsigned char debug)

  Enables debug output from the CoreDX DDS run-time license system.
- 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.
- char * CoreDX_DDS_get_lib_version (void)

  Returns the version string associated with the CoreDX DDS library.
- DDS_ReturnCode_t CoreDX_DDS_get_lib_version_string (char *vstring_buf, int buf_len)

  Returns the version string associated with the CoreDX DDS library.

2.5.1 Detailed Description

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

2.5.2 Friends And Related Function Documentation

2.5.2.1 char * CoreDX_DDS_get_lib_version (void) [related]

Returns the version string associated with the CoreDX DDS library.

Deprecated Use CoreDX_DDS_get_lib_version_string() instead.

Routine allocates a buffer to hold the version string. Caller must free the buffer.

2.5.2.2 DDS_ReturnCode_t CoreDX_DDS_get_lib_version_string (char * *vstring_buf,* int *buf_len*) [related]

Returns the version string associated with the CoreDX DDS library.

Caller must provide a buffer to hold the version string 'vstring_buf' If the paramter 'buf_len' indicates that buffer is too small to hold the version string, then DDS_RETCODE_BAD_PARAMETER is returned. If the 'vstring_buf' is NULL, then DDS_RETCODE_BAD_PARAMETER is returned.

2.5.2.3 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 DDS to use the default qos settings held in the DomainParticipantFactory.
This routine will return **NULL** if it fails to create a DomainParticipant.

Note

: deprectated...

2.5.2.4 DDS_DomainParticipant DDS_DomainParticipantFactory_create_participant_ex (DDS_DomainParticipantFactory *dpf,* 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 DDS to use the default qos settings held in the DomainParticipantFactory.
This routine will return **NULL** if it fails to create a DomainParticipant.

2.5.2.5 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.)

2.5.2.6 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.

2.5.2.7 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.

2.5.2.8 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.

2.5.2.9 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 '>'.

2.5.2.10 DDS_ReturnCode_t DDS_DomainParticipantFactory_set_license_debug (unsigned char *debug*) [related]

Enables debug output from the CoreDX DDS run-time license system.

This routine will configure CoreDX DDS to generate debug information in processing the license at run-time. The 'deubg' parameter may be either: 'nonzero' to enable debug output or 'zero' to disable debug ouput.

2.5.2.11 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.

2.6 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 CoreDX_DDS_heap_init (void *heap_ptr, uint32_t size)

  Provide a block of memory for CoreDX DDS to use.
- 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**.*
- DDS_ReturnCode_t DDS_DomainParticipant_get_discovered_publication_data (DDS_DomainParticipant dp, DDS_PublicationBuiltinTopicData *publication_data, const DDS_InstanceHandle_t publication_handle)

  *This operation returns data that describes a particular known DataWriter identified by **publication_handle**.*
- DDS_ReturnCode_t DDS_DomainParticipant_get_discovered_subscription_data (DDS_DomainParticipant dp, DDS_SubscriptionBuiltinTopicData *subscription_data, const DDS_InstanceHandle_t subscription_handle)

  *This operation returns data that describes a particular known DataReader identified by **subscription_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 provided version information against the version of the running CoreDX DDS library.
- int DDS_DomainParticipant_validate_version (DDS_DomainParticipant dp, const char *source, const char *major, const char *minor, const char *patch)

  Tests the provided version information against the version of the running CoreDX DDS library.
- DDS_ReturnCode_t DDS_DomainParticipant_do_work (DDS_DomainParticipant dp, const 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, const DDS_Duration_t *max_wait)

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

- void DDS_DomainParticipant_print_stats (DDS_DomainParticipant dp)

  Prints simple summary debug stats to stdout for each Reader and Writer.
- DDS_ReturnCode_t DDS_DomainParticipant_add_transport (DDS_DomainParticipant dp, struct CoreDX_Transport *transport)

  Adds the provided transport to the DomainParticipant.
- DDS_ReturnCode_t DDS_DomainParticipant_get_guid (DDS_DomainParticipant dp, DDS_GUID_t *guid)

  Access the GUID which uniquely identifies this participant.
- DDS_ReturnCode_t DDS_DomainParticipant_create_typesupport_from_typecode (DDS_DomainParticipant dp, DDS_TypecodeQosPolicy *tc_qos, DDS_TypeSupport *ts)

  Construct a TypeSupport to handle the type identified by the provided TypecodeQosPolicy.
- DDS_ReturnCode_t DDS_DomainParticipant_create_typesupport_from_typeobj (DDS_DomainParticipant dp, DDS_TypecodeQosPolicy *tc_qos, DDS_TypeSupport *ts)

  Construct a TypeSupport to handle the type identified by the provided TypecodeQosPolicy.

2.6.1 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.

2.6.2 Friends And Related Function Documentation

2.6.2.1 DDS_ReturnCode_t CoreDX_DDS_heap_init (void * *heap_ptr*, uint32_t *size*) [related]

Provide a block of memory for CoreDX DDS to use.

On CoreDX DDS builds configured to **not** use Heap memory allocation routines from the Operating System (common for embedded or Safety Critical environments), this routine is used to assign a block of memory for CoreDX DDS usage. All memory required by CoreDX DDS will be constrained to this block of memory, and no other memory resources will be used.

Note

It is required that the heap be initialized prior to making any other calls to the CoreDX DDS API.

Return values

DDS_RETCODE_OK	on success
DDS_RETCODE_BAD_PARAMETER	if 'size == 0' or 'heap_ptr == NULL'
DDS_RETCODE_UNSUPPORTED	if the build is configured to use the Heap from the OS.

2.6.2.2 DDS_ReturnCode_t DDS_DomainParticipant_add_transport (DDS_DomainParticipant *dp*, struct CoreDX_Transport * *transport*) [related]

Adds the provided transport to the DomainParticipant.

The transport is registered with the DomainParticipant, and will be used for communication. If no transports are manually added, then the DomainParticipant will automatically add the default UDP transport. The DomainParticipant takes

ownership of the provided transport; when the DomainParticipant is deleted, it will release the resources of all added transports.

Note

Transports can be installed only before a DomainParticipant is enabled.

2.6.2.3 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'.

2.6.2.4 int DDS_DomainParticipant_check_version (DDS_DomainParticipant *dp,* const char * *major,* const char * *minor,* const char * *patch*) [related]

Tests the provided version information against the version of the running CoreDX DDS 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.

2.6.2.5 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.

2.6.2.6 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.

2.6.2.7 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.

2.6.2.8 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.

2.6.2.9 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.

2.6.2.10 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().

2.6.2.11 DDS_ReturnCode_t DDS_DomainParticipant_create_typesupport_from_typecode (DDS_DomainParticipant *dp*, DDS_TypecodeQosPolicy * *tc_qos*, DDS_TypeSupport * *ts*) [related]

Construct a TypeSupport to handle the type identified by the provided TypecodeQosPolicy.

A discovered publication or subscription may include type information in the TypecodeQosPolicy. This type information can be used to create a TypeSupport for use by CoreDX DDS. The application can use this TypeSupport to register a type, create a Topic, and create DataReaders and DataWriters. The TypecodeQosPolicy can hold either a 'TypeCode' or a 'TypeObject' type representation.

This routine can be successful only if the TypecodeQosPolicy holds a 'TypeCode'; otherwise, it returns DDS_RETCODE_BAD_PARAMETER.

2.6.2.12 DDS_ReturnCode_t DDS_DomainParticipant_create_typesupport_from_typeobj (DDS_DomainParticipant *dp*, DDS_TypecodeQosPolicy * *tc_qos*, DDS_TypeSupport * *ts*) [related]

Construct a TypeSupport to handle the type identified by the provided TypecodeQosPolicy.

A discovered publication or subscription may include type information in the TypecodeQosPolicy. This type information can be used to create a TypeSupport for use by CoreDX DDS. The application can use this TypeSupport to register a type, create a Topic, and create DataReaders and DataWriters. The TypecodeQosPolicy can hold either a 'TypeCode' or a 'TypeObject' type representation.

This routine can be successful only if the TypecodeQosPolicy holds a 'TypeObject'; otherwise, it returns DDS_RETCODE_BAD_PARAMETER.

2.6.2.13 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.

2.6.2.14 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_

The Topic must be owned by DomainPartipant **dp**; otherwise DDS_RETCODE_PRECONDITION_NOT_MET will be returned.

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

Not Yet Supported This operation is not implemented yet.

2.6.2.16 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_**

2.6.2.17 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_**

2.6.2.18 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_

The Topic must be owned by DomainPartipant **dp**; otherwise DDS_RETCODE_PRECONDITION_NOT_MET will be returned.

2.6.2.19 DDS_ReturnCode_t DDS_DomainParticipant_do_work (DDS_DomainParticipant *dp,* const 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.

2.6.2.20 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].

Return values

DDS_RETCODE_OK	if the Participant is enabled OK.
DDS_RETCODE_ERROR	if **all** installed transports failled to initialize successfully.
DDS_RETCODE_BAD_PARAMETER	if a null 'dp' value is passed in.

2.6.2.21 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().

2.6.2.22 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.

2.6.2.23 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.

2.6.2.24 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.

2.6.2.25 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.

2.6.2.26 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.
To reclaim any memory returned in the participant_data paramter, call DDS_DCPSParticipant_clear(participant_data).

2.6.2.27 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.

2.6.2.28 DDS_ReturnCode_t DDS_DomainParticipant_get_discovered_publication_data (DDS_DomainParticipant *dp,* DDS_PublicationBuiltinTopicData * *publication_data,* const DDS_InstanceHandle_t *publication_handle*) [related]

This operation returns data that describes a particular known 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 known DataWriter, this routine will return DDS_RETCODE_PRECONDITION_NOT_MET.
To reclaim any memory returned in the 'publication_data' parameter, call DDS_DCPSPublication_clear(publication_data).

2.6.2.29 DDS_ReturnCode_t DDS_DomainParticipant_get_discovered_subscription_data (DDS_DomainParticipant *dp*, DDS_SubscriptionBuiltinTopicData * *subscription_data*, const DDS_InstanceHandle_t *subscription_handle*) [related]

This operation returns data that describes a particular known 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 known DataReader, this routine will return DDS_RETCODE_PRECONDITION_NOT_MET. To reclaim any memory returned in the 'subscription_data' parameter, call DDS_DCPSSubscription_clear(subscription_data).

2.6.2.30 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.

Not Yet Supported

2.6.2.31 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.

2.6.2.32 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.

2.6.2.33 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.

2.6.2.34 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.

2.6.2.35 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.

2.6.2.36 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.

2.6.2.37 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.

2.6.2.38 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.

2.6.2.39 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.

2.6.2.40 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.

2.6.2.41 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.

2.6.2.42 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.

2.6.2.43 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.

2.6.2.44 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.

2.6.2.45 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.

2.6.2.46 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.

2.6.2.47 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.

2.6.2.48 int DDS_DomainParticipant_validate_version (DDS_DomainParticipant *dp*, const char * *source*, const char * *major*, const char * *minor*, const char * *patch*) [related]

Tests the provided version information against the version of the running CoreDX DDS library.

Returns the same values as DDS_DomainParticpant_check_version. Also will print an error to standard error if the provided version does not match the running library version. The printed error message will contain the string in 'source' for reference.

2.7 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 (void)

  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**.*

2.7.1 Detailed Description

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

2.7.2 Friends And Related Function Documentation

2.7.2.1 DDS_GuardCondition DDS_GuardCondition__alloc (void) [related]

This routine allocates a DDS_GuardCondition.

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

2.7.2.2 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().

2.7.2.3 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.

2.7.2.4 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.]

2.8 DDS_MemberDescriptor Struct Reference

A DDS_MemberDescriptor comprises the state of a DDS_DynamicTypeMember.

Data Fields

- DDS_ObjectName name

  the 'name' of this type
- DDS_MemberId id

  If this member belongs to an aggregated type, this property indicates the member's ID.
- DDS_DynamicType type

  This property indicates the type of the member's value.
- char * default_value

  This property provides the member's default value in string form.
- unsigned int index

  This property indicates the order of definition of this member within its type, relative to the type's other members.
- DDS_UnionCaseLabelSeq label

  If the type to which the member belongs is a union, this property indicates the case labels that apply to this member.
- unsigned char default_label

  For this descriptor to be consistent, this property must be true if this descriptor identifies the default member of a union type or false if not.

Related Functions

(Note that these are not member functions.)

- DDS_ReturnCode_t DDS_MemberDescriptor_copy_from (DDS_MemberDescriptor *to, const DDS_MemberDescriptor *from)

  Overwrite the contents of the 'to' descriptor with those of the provided 'from' descriptor.
- unsigned char DDS_MemberDescriptor_equals (DDS_MemberDescriptor *md, const DDS_MemberDescriptor *other)

  Compare two DDS_TypeDescriptor's.
- unsigned char DDS_MemberDescriptor_is_consistent (DDS_MemberDescriptor *md)

  Indicates whether the states of all of this descriptor's properties are consistent.

2.8.1 Detailed Description

A DDS_MemberDescriptor comprises the state of a DDS_DynamicTypeMember.

2.8.2 Field Documentation

2.8.2.1 unsigned char DDS_MemberDescriptor::default_label

For this descriptor to be consistent, this property must be true if this descriptor identifies the default member of a union type or false if not.

A default union member may have additional explicit labels (indicated in the label property), but these are semantically irrelevant, as the default member would be in effect or not regardless of their presence or absence.

2.8.2.2 char* DDS_MemberDescriptor::default_value

This property provides the member's default value in string form.

A string representation of a data value is considered well formed if it would be a valid IDL literal of the corresponding type with the following qualifications:

- String and character literals shall not be surrounded by quotation characters (‘"’ or ‘'’).
- All expressions shall be fully evaluated such that no operators or other non-literal characters occur in the value. For example, "5" is a well-formed string representation of the integer quantity five, but "2 + ENUM_VALUE_THREE" is not.

A nil or empty string indicates that the member takes the "default default" value for its type. This rule shall always be used when the member is of a type for which IDL provides no syntax to express a literal value (for example, structures or maps) and may be used for any other type.

2.8.2.3 DDS_MemberId DDS_MemberDescriptor::id

If this member belongs to an aggregated type, this property indicates the member's ID.

When a descriptor is used to add a new member to a type, this property may be set to MEMBER_ID_INVALID; in that case, the implementation shall select an ID for the new member that is one more than the current maximum member ID in the type. If the value of this property is not MEMBER_ID_INVALID, it must be set to a value within a legal range.

When a descriptor is retrieved from an existing member, this property shall reflect the actual ID of the member. It shall therefore not be MEMBER_ID_INVALID, and it shall fall within a legal range.

If this member does not belong to an aggregated type, this property must be MEMBER_ID_INVALID, or the descriptor is not consistent.

2.8.2.4 unsigned int DDS_MemberDescriptor::index

This property indicates the order of definition of this member within its type, relative to the type's other members.

The first member shall have index zero, the next index one, and so on.

When a descriptor is used to add a new member to a type, any value greater than the current largest index value in the type shall be taken to indicate that the new member will become the last member, whatever the index; member indices within a type shall not be discontiguous.

Alternatively, if this property is set to an index at which a member already exists, that member and all those after it shall be shifted by a single index value to make room for the new member.

When a descriptor is retrieved from an existing member, this property shall always reflect the actual index at which the member exists.

2.8.2.5 DDS_UnionCaseLabelSeq DDS_MemberDescriptor::label

If the type to which the member belongs is a union, this property indicates the case labels that apply to this member.

If default_label is false, it must not be empty. In addition, no two members of the same union can specify the same label value.

If the type to which the member belongs is not a union, this property's value must be empty to be consistent.

2.8.2.6 DDS_DynamicType DDS_MemberDescriptor::type

This property indicates the type of the member's value.

It must not be nil, and it must indicate a type that can legally type a member according to the Type System Model.

2.9 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.

2.9.1 Detailed Description

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

Not Yet Supported CoreDX does not yet implement MultiTopics.

2.10 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_ReturnCode_t DDS_Publisher_wait_for_acknowledgments (DDS_Publisher p, const DDS_Duration_t *max_wait)

  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****.*

2.10.1 Detailed Description

The DDS_Publisher configures, creates, manages and destroys DDS_DataWriters.

2.10.2 Friends And Related Function Documentation

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

Not Yet Supported This operation is not yet implemented.

2.10.2.2 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.

2.10.2.3 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**.

2.10.2.4 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.

2.10.2.5 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.

2.10.2.6 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].

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

Not Yet Supported This operation is not yet implemented.

2.10.2.8 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.

2.10.2.9 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.

2.10.2.10 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.

2.10.2.11 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)**.

2.10.2.12 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.

2.10.2.13 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.

2.10.2.14 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**.

2.10.2.15 DDS_ReturnCode_t DDS_Publisher_resume_publications (DDS_Publisher *p*) [related]

Not Yet Supported This operation is not yet implemented.

2.10.2.16 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.

2.10.2.17 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.

2.10.2.18 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.

2.10.2.19 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.

2.10.2.20 DDS_ReturnCode_t DDS_Publisher_suspend_publications (DDS_Publisher *p*) [related]

Not Yet Supported This operation is not yet implemented.

2.10.2.21 DDS_ReturnCode_t DDS_Publisher_wait_for_acknowledgments (DDS_Publisher *p,* const DDS_Duration_t * *max_wait*) [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'

2.11 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.

2.11.1 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 DDS does not yet implement QueryConditions as a trigger for a WaitSet.

2.11.2 Friends And Related Function Documentation

2.11.2.1 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.

2.11.2.2 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()

2.11.2.3 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.

2.11.2.4 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()

2.12 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.

2.12.1 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().

2.12.2 Friends And Related Function Documentation

2.12.2.1 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.

2.13 DDS_SampleInfo Struct Reference

The SampleInfo structure contains information associated with each sample.

Data Fields

- 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.

2.13.1 Detailed Description

The SampleInfo structure contains information associated with each sample.

2.13.2 Field Documentation

2.13.2.1 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.

2.13.2.2 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.

2.13.2.3 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) ...
```

2.13.2.4 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.

2.13.2.5 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).

2.13.2.6 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

2.14 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.

2.14.1 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.

2.14.2 Friends And Related Function Documentation

2.14.2.1 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.

2.14.2.2 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.

2.14.2.3 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.

2.15 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**.*

2.15.1 Detailed Description

The DDS_Subscriber configures, creates, manages and destroys DDS_DataReaders.

2.15.2 Friends And Related Function Documentation

2.15.2.1 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.]

2.15.2.2 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.

2.15.2.3 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**.

2.15.2.4 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.

2.15.2.5 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.

2.15.2.6 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].

2.15.2.7 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.

2.15.2.8 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.

2.15.2.9 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.

2.15.2.10 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.

2.15.2.11 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)**.

2.15.2.12 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.

2.15.2.13 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.

2.15.2.14 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.

2.15.2.15 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.

2.15.2.16 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.

2.15.2.17 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.

2.15.2.18 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.

2.16 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.

2.16.1 Detailed Description

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

2.16.2 Friends And Related Function Documentation

2.16.2.1 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.

2.16.2.2 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.

2.17 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.

2.17.1 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.]

2.17.2 Friends And Related Function Documentation

2.17.2.1 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].

2.17.2.2 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.

2.17.2.3 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.

2.17.2.4 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.

2.17.2.5 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)**.

2.17.2.6 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**.

2.17.2.7 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.

2.17.2.8 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.

2.17.2.9 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.

2.17.2.10 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.

2.18 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 (void)

  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.

2.18.1 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().

2.18.2 Friends And Related Function Documentation

2.18.2.1 DDS_WaitSet DDS_WaitSet__alloc (void) [related]

Convenience routine to allocate a DDS_WaitSet.

Allocates memory which should be returned via DDS_WaitSet__free().

2.18.2.2 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.]

2.18.2.3 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.

2.18.2.4 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.

2.18.2.5 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.

2.18.2.6 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.

Chapter 3

Entity QoS Collections

3.1 DDS_DataReaderQos Struct Reference

Structure that holds DDS_DataReader Quality of Service policies.

Data Fields

- 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.
- DDS_DataRepresentationQosPolicy representation

  Informs DataWriter(s) of the data representation(s) supported by this Reader. This must be consistent with the TypeSupport associated with the Reader's Topic.

- DDS_TypeConsistencyEnforcementQosPolicy type_consistency

  Influences the algorithm that matches DataReaders and DataWriters based on their data type compatibility.
- DDS_RpcQosPolicy rpc

  Configure RPC relevant settings: instance_name, related_entity, and topic_aliases.
- CoreDX_EntityNameQosPolicy entity_name

  entity name
- CoreDX_LoggingQosPolicy logging

  logging
- CoreDX_RTPSReaderQosPolicy rtps_reader

  rtps_reader configuration

3.1.1 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()

3.2 DDS_DataWriterQos Struct Reference

Structure that holds DDS_DataWriter Quality of Service policies.

Data Fields

- 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.
- DDS_DataRepresentationQosPolicy representation

  Informs DataReader(s) of the single data representation supported by this Writer. This must be consistent with the TypeSupport associated with the Writer's Topic.
- DDS_RpcQosPolicy rpc

  Configure RPC relevant settings: instance_name, related_entity, and topic_aliases.
- CoreDX_EntityNameQosPolicy entity_name

  entity name
- CoreDX_LoggingQosPolicy logging

  logging
- CoreDX_RTPSWriterQosPolicy rtps_writer

  rtps_writer configuration

3.2.1 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()

3.3 DDS_DomainParticipantFactoryQos Struct Reference

Structure that holds DDS_DomainParticipantFactory Quality of Service policies.

Data Fields

- DDS_EntityFactoryQosPolicy entity_factory

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

3.3.1 Detailed Description

Structure that holds DDS_DomainParticipantFactory Quality of Service policies.

See also

- DDS_DomainParticipantFactory_set_qos()
- DDS_DomainParticipantFactory_get_qos()

3.4 DDS_DomainParticipantQos Struct Reference

Structure that holds DDS_DomainParticipant Quality of Service policies.

Data Fields

- 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_EntityNameQosPolicy entity_name

  entity name
- CoreDX_LoggingQosPolicy logging

  logging
- 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.
- DDS_PropertyQosPolicy properties

  Additional name:value pair properties (if propagate=true, included in discovery)

3.4.1 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()

3.5 DDS_PublisherQos Struct Reference

Structure that holds DDS_Publisher Quality of Service policies.

Data Fields

- 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.
- CoreDX_EntityNameQosPolicy entity_name
  entity name
- CoreDX_LoggingQosPolicy logging
  logging

3.5.1 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()

3.5.2 Field Documentation

3.5.2.1 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.

3.5.2.2 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()

3.6 DDS_SubscriberQos Struct Reference

Structure that holds DDS_Subscriber Quality of Service policies.

Data Fields

- 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.
- CoreDX_EntityNameQosPolicy entity_name
  entity name
- CoreDX_LoggingQosPolicy logging
  logging

3.6.1 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()

3.6.2 Field Documentation

3.6.2.1 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.

3.6.2.2 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()

3.7 DDS_TopicQos Struct Reference

Structure that holds DDS_Topic Quality of Service policies.

Data Fields

- 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.
- DDS_DataRepresentationQosPolicy representation

  The data representation(s) supported by Writer(s) and Reader(s) of this topic. The first entry in the 'value' sequence corresponds to the Writer(s), all entries apply to Reader(s).
- CoreDX_EntityNameQosPolicy entity_name

  entity_name
- CoreDX_LoggingQosPolicy logging

  logging

3.7.1 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()

Chapter 4

QoS Policies

4.1 CoreDX_DiscoveryQosPolicy Struct Reference

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

Data Fields

- 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

  Multicast DiscoveredParticipantData each period.
- DDS_Duration_t dpd_lease_duration

  How long we consider a discoverd Participant to be alive without hearing from them.
- DDS_Duration_t heartbeat_period

  The heartbeat_period to configure for builtin writers.
- DDS_Duration_t nack_response_delay

  The nack_response_delay to configure for builtin writers.
- DDS_Duration_t nack_suppress_delay

  The nack_suppress_delay to configure for builtin writers.
- uint32_t min_buffer_size

  minimum tx buffer size in bytes for builtin writers
- uint32_t max_buffer_size

  maximum tx buffer size in bytes for builtin writers
- DDS_Duration_t heartbeat_response_delay

  The heartbeat_response_delay to configure for builtin readers.
- unsigned char send_initial_nack

  send a ACKNACK immediately after matching with new Writer.
- unsigned char send_msglen_submsg

  include 'msglen' submessage in RTPS header

4.1.1 Detailed Description

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

4.2 CoreDX_EntityNameQosPolicy Struct Reference

Data Fields

- char value [COREDX_ENTITY_NAME_MAX]

  string to use as entity name

4.2.1 Detailed Description

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

4.3 CoreDX_LoggingQosPolicy Struct Reference

Controls the amount and kind of information that is logged.

Data Fields

- uint32_t flags

  flags to control logging output

4.3.1 Detailed Description

Controls the amount and kind of information that is logged.

The application must be linked with the instrumented library in order to enable logging with this LoggingQosPolicy.

The level of log output is controlled by a series of bitmapped flags. Each class of log message is enabled by setting a flag to 1, and is disabled by 0.

Refer to include/dds/coredx_logging.h for the numeric values for the flags

4.4 CoreDX_PeerParticipantQosPolicy Struct Reference

Configures a list of DomainParticipant peers to attempt communication with.

Data Fields

- CoreDX_ParticipantLocatorSeq value
- unsigned char strict_match

4.4.1 Detailed Description

Configures a list of DomainParticipant peers to attempt communication with.

If the value list is empty, then default Discovery is used.

4.4.2 Field Documentation

4.4.2.1 unsigned char CoreDX_PeerParticipantQosPolicy::strict_match

restrict matching to those peers listed (default is 0/FALSE)

4.4.2.2 CoreDX_ParticipantLocatorSeq CoreDX_PeerParticipantQosPolicy::value

sequence of Peer Locators with which we should attempt communication

4.5 CoreDX_RTPSReaderQosPolicy Struct Reference

QoS Policy for configuring aspects of the RTPS Reader Protocol.

Data Fields

- DDS_Duration_t heartbeat_response_delay
  ammount of time allowed for responding to a heartbeat
- unsigned char accept_batch_msg
- unsigned char send_typecode
- unsigned char send_initial_nack
- uint32_t precache_max_samples

4.5.1 Detailed Description

QoS Policy for configuring aspects of the RTPS Reader Protocol.

4.5.2 Field Documentation

4.5.2.1 unsigned char CoreDX_RTPSReaderQosPolicy::accept_batch_msg

allow writers to use the 'BATCH' RTPS message

4.5.2.2 uint32_t CoreDX_RTPSReaderQosPolicy::precache_max_samples

Maximum number of samples held in pre-cache [only for RELIABLE readers].

4.5.2.3 unsigned char CoreDX_RTPSReaderQosPolicy::send_initial_nack

send a ACKNACK immediately after matching with new Writer.

4.5.2.4 unsigned char CoreDX_RTPSReaderQosPolicy::send_typecode

send 'typecode' information for associated data type.

4.6 CoreDX_RTPSWriterQosPolicy Struct Reference

QoS Policy for configuring aspects of the RTPS Writer Protocol.

Data Fields

- DDS_Duration_t heartbeat_period

  period to transmit heartbeat messages if required
- DDS_Duration_t nack_response_delay

  ammount of time allowed for responding to NACKs
- DDS_Duration_t nack_suppress_delay

  ammount of time to ignore NACKs after a repair
- DDS_Duration_t ack_deadline
- uint32_t min_buffer_size
- uint32_t max_buffer_size
- unsigned char apply_filters
- unsigned char enable_batch_msg
- unsigned char send_typecode

4.6.1 Detailed Description

QoS Policy for configuring aspects of the RTPS Writer Protocol.

4.6.2 Field Documentation

4.6.2.1 DDS_Duration_t CoreDX_RTPSWriterQosPolicy::ack_deadline

after which a realiable reader will be considered unresponsive

4.6.2.2 unsigned char CoreDX_RTPSWriterQosPolicy::apply_filters

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

4.6.2.3 unsigned char CoreDX_RTPSWriterQosPolicy::enable_batch_msg

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

4.6.2.4 uint32_t CoreDX_RTPSWriterQosPolicy::max_buffer_size

max size in bytes of written data

4.6.2.5 uint32_t CoreDX_RTPSWriterQosPolicy::min_buffer_size

min size in bytes of written data

4.6.2.6 unsigned char CoreDX_RTPSWriterQosPolicy::send_typecode

send 'typecode' information for associated data type.

4.7 CoreDX_ThreadModelQosPolicy Struct Reference

QoS Policy for configuring the threading behavior of the DomainParticipant.

Data Fields

- unsigned char use_threads

  should the DomainParticipant use threads for internal work
- unsigned char create_listener_thread

4.7.1 Detailed Description

QoS Policy for configuring the threading behavior of the DomainParticipant.

4.7.2 Field Documentation

4.7.2.1 unsigned char CoreDX_ThreadModelQosPolicy::create_listener_thread

create a distinct thread for application Listener callbacks

4.8 DDS_DataRepresentationQosPolicy Struct Reference

Describes the data representation used by a topic.

Data Fields

- DDS_DataRepresentationIdSeq value
  data representation

4.8.1 Detailed Description

Describes the data representation used by a topic.

4.9 DDS_DeadlineQosPolicy Struct Reference

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

Data Fields

- struct DDS_Duration_t period

  the maximum interval between instance updates.

4.9.1 Detailed Description

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

DataWriters specify that each data instance will be updated at least once every 'period'. A DataReader requests that the Writer commit to updating each instance at least once every 'period'.

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

If a DataWriter fails to meet the update period, then the DataWriter is notified by a call to the offered_deadline_missed listener, and the DataReader is notified by a call to the requested_deadline_missed listener.

See also

DataReaderListener::on_requested_deadline_missed(DataReader the_reader, RequestedDeadlineMissedStatus status) on_requested_deadline_missed
DataWriterListener::on_offered_deadline_missed(DataWriter writer, OfferedDeadlineMissedStatus status) on_offered_deadline_missed

4.10 DDS_DestinationOrderQosPolicy Struct Reference

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

Data Fields

- DDS_DestinationOrderQosPolicyKind kind

  the kind of ordering applied to samples

4.10.1 Detailed Description

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

Can be **BY_RECEPTION_TIMESTAMP** or **BY_SOURCE_TIMESTAMP**. The samples are ordered either based on their order of reception, or by the order of timestamps generated by the source (DataWriter).

CoreDX DDS currently implements only BY_RECEPTION_TIMESTAMP.

4.11 DDS_DurabilityQosPolicy Struct Reference

The DurabilityQosPolicy controls the durablity of data.

Data Fields

- DDS_DurabilityQosPolicyKind kind

  the durability kind

4.11.1 Detailed Description

The DurabilityQosPolicy controls the durablity of data.

The DDS API identifies several degrees of data durability.

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

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

CoreDX DDS currently supports VOLATILE and TRANSIENT_LOCAL.

4.12 DDS_DurabilityServiceQosPolicy Struct Reference

Data Fields

- struct DDS_Duration_t service_cleanup_delay

  service cleanup delay
- DDS_HistoryQosPolicyKind history_kind

  history kind
- int history_depth

  history depth
- int max_samples

  maximum samples limit for the service
- int max_instances

  maximum instances limit for the service
- int max_samples_per_instance

  maximum samples per instance limit for the service

4.12.1 Detailed Description

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

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

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

CoreDX DDS currently supports VOLATILE and TRANSIENT_LOCAL.

4.13 DDS_EntityFactoryQosPolicy Struct Reference

Data Fields

- unsigned char autoenable_created_entities

  should created entities be automatically enabled

4.13.1 Detailed Description

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

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

4.14 DDS_GroupDataQosPolicy Struct Reference

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

Data Fields

- DDS_OctetSeq value
  sequence of bytes

4.14.1 Detailed Description

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

The value of the GROUP_DATA is propagated and made available to applications through the built-in topics.

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

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

4.15 DDS_HistoryQosPolicy Struct Reference

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

Data Fields

- DDS_HistoryQosPolicyKind kind

  history kind
- int depth

  history depth

4.15.1 Detailed Description

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

The history kind can be set to KEEP_ALL_HISTORY_QOS or KEEP_LAST_HISTORY_QOS.

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

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

This QoS policy must be consistent with the ResourceLimits policy.

See also

ResourceLimitsQosPolicy

4.16 DDS_LatencyBudgetQosPolicy Struct Reference

Specifies allowable latency.

Data Fields

- struct DDS_Duration_t duration

  the maximum delay before processing.

4.16.1 Detailed Description

Specifies allowable latency.

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

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

4.17 DDS_LifespanQosPolicy Struct Reference

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

Data Fields

- struct DDS_Duration_t duration

  lifespan duration

4.17.1 Detailed Description

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

The default value of the lifespan duration is infinite.

4.18 DDS_LivelinessQosPolicy Struct Reference

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

Data Fields

- DDS_LivelinessQosPolicyKind kind

  the kind of liveliness mechanism
- struct DDS_Duration_t lease_duration

  the duration of the liveliness lease

4.18.1 Detailed Description

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

The QoS defines the kind of liveliness (how liveliness is maintained) as well as a 'lease_duration' which specifies an interval for checking liveliness. The liveliness kind can be one of the following:

- AUTOMATIC
- MANUAL_BY_PARTICIPANT
- MANUAL_BY_TOPIC

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

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

See also

DataReaderListener::on_liveliness_changed(DataReader the_reader, LivelinessChangedStatus DDS Status Structures) on_liveliness_changed
DataWriterListener::on_liveliness_lost(DataWriter writer, LivelinessLostStatus DDS Status Structures) on_liveliness_lost

4.19 DDS_OwnershipQosPolicy Struct Reference

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

Data Fields

- DDS_OwnershipQosPolicyKind kind

  the ownership kind

4.19.1 Detailed Description

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

4.20 DDS_OwnershipStrengthQosPolicy Struct Reference

Defines the strength, or priority, of a Writer. The strength is used to determine ownership in the case of EXCLUSIVE_OWNERSHIP_QOS. When multiple writers publish data about the same instance, the **stronger** writer is considered the owner, and data from other writers is not delivered to the reader.

Data Fields

- int value

  the ownership strength value

4.20.1 Detailed Description

Defines the strength, or priority, of a Writer. The strength is used to determine ownership in the case of EXCLUSIVE_OWNERSHIP_QOS. When multiple writers publish data about the same instance, the **stronger** writer is considered the owner, and data from other writers is not delivered to the reader.

4.21 DDS_PartitionQosPolicy Struct Reference

Defines a logical data partition.

Data Fields

- DDS_StringSeq name

  sequence of partition names

4.21.1 Detailed Description

Defines a logical data partition.

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

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

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

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

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

4.22 DDS_PresentationQosPolicy Struct Reference

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

Data Fields

- DDS_PresentationQosPolicyAccessScopeKind access_scope

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

  Determines if coherent access is supported within the defined 'scope'.
- unsigned char ordered_access

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

4.22.1 Detailed Description

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

4.23 DDS_PropertyQosPolicy Struct Reference

Data Fields

- DDS_PropertySeq value

  sequence of Property_t's

4.23.1 Detailed Description

The PropertyQosPolicy supports adding arbitrary name-value pairs. This can be used to configure non-standard properties in a portable fashion. In particular, this can be used to configure aspects of Security Plugin implementations.

4.24 DDS_ReaderDataLifecycleQosPolicy Struct Reference

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

Data Fields

- DDS_Duration_t autopurge_nowriter_samples_delay

  delay after which it is ok to purge samples from instances with no writer(s)
- DDS_Duration_t autopurge_disposed_samples_delay

  delay after which it is ok to purge samples from instances that are disposed

4.24.1 Detailed Description

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

autopurge_nowriter_samples_delay indicates how long the DataReader will maintain instance samples that have instance_state NOT_ALIVE_NO_WRITERS.

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

4.25 DDS_ReliabilityQosPolicy Struct Reference

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

Data Fields

- DDS_ReliabilityQosPolicyKind kind

  reliability kind
- struct DDS_Duration_t max_blocking_time

  maximum blocking time

4.25.1 Detailed Description

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

A kind of BEST_EFFORT_RELIABILITY_QOS indicates that the middleware does not need to retry delivery of data samples.

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

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

See also

HistoryQosPolicy
ResourceLimitsQosPolicy
DurabilityQosPolicy

4.26 DDS_ResourceLimitsQosPolicy Struct Reference

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

Data Fields

- int max_samples

  maximum samples allowed in the cache
- int max_instances

  maximum instances allowed in the cache
- int max_samples_per_instance

  maximum samples per instance allowed in the cache
- unsigned char preallocate_samples

  Request that the Reader or Writer pre-allocate the specified number of samples (if max_samples is not unlimited). CoreDX DDS Extension.
- unsigned char preallocate_instances

  Request that the Reader or Writer pre-allocate the specified number of instances (if max_instances is not unlimited). CoreDX DDS Extension.

4.26.1 Detailed Description

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

See also

ReliabilityQosPolicy
HistoryQosPolicy

4.27 DDS_RpcQosPolicy Struct Reference

Augment a DataWriter or DataReader with RPC specific information.

Data Fields

- DDS_RpcQosPolicy_fixedstring255_t service_instance_name

  the name of the service instance
- struct DDS_GUID_t related_entity_guid

  guid of the related entity (request / reply)
- DDS_StringSeq topic_aliases

  a sequence of aliases to support derived interfaces

4.27.1 Detailed Description

Augment a DataWriter or DataReader with RPC specific information.

4.28 DDS_TimeBasedFilterQosPolicy Struct Reference

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

Data Fields

- struct DDS_Duration_t minimum_separation

  the minimum separation between instance updates

4.28.1 Detailed Description

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

It is inconsistent for a DataReader to have a minimum_separation larger than its DEADLINE period. By default minimum_separation=0 indicating that the DataReader should be sent all values.

4.29 DDS_TopicDataQosPolicy Struct Reference

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

Data Fields

- DDS_OctetSeq value
  sequence of bytes

4.29.1 Detailed Description

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

The value of the TOPIC_DATA is propagated and made available to applications through the built-in topics.

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

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

4.30 DDS_TransportPriorityQosPolicy Struct Reference

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

Data Fields

- int value

  priority value

4.30.1 Detailed Description

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

4.31 DDS_TypecodeQosPolicy Struct Reference

Typecode representing the datatype a DataReader reads or a DataWriter writes.

Data Fields

- DDS_OctetSeq value

  sequence of bytes representing the TypeCode for the data type
- unsigned char encoding

  encoding of the typecode bytes (little endian / big endian

4.31.1 Detailed Description

Typecode representing the datatype a DataReader reads or a DataWriter writes.

4.32 DDS_TypeConsistencyEnforcementQosPolicy Struct Reference

rules for determining type consistency

Data Fields

- DDS_TypeConsistencyKind kind
  type consistency kind

4.32.1 Detailed Description

rules for determining type consistency

The Type Consistency Enforcement QoS Policy defines the rules for determining whether the type used to publish a given data stream is consistent with that used to subscribe to it. It applies to DataReaders.

4.33 DDS_UserDataQosPolicy Struct Reference

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

Data Fields

- DDS_OctetSeq value
  sequence of bytes

4.33.1 Detailed Description

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

The value of the USER_DATA is propagated and made available to applications through the built-in topics.

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

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

4.34 DDS_WriterDataLifecycleQosPolicy Struct Reference

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

Data Fields

- unsigned char autodispose_unregistered_instances

  should the writer dispose instances that are unregistered

4.34.1 Detailed Description

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

Chapter 5

DDS Listeners

5.1 DDS_DataReaderListener_cd Struct Reference

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

Data Fields

- 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)

5.1.1 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.

5.1.2 Field Documentation

5.1.2.1 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 DDS middleware 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.

5.1.2.2 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 DDS middleware 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.

5.1.2.3 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 DDS middleware 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.

5.1.2.4 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 DDS middleware 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.

5.1.2.5 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 DDS middleware 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.

5.1.2.6 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 DDS middleware detects that a sample has been rejected.

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

5.1.2.7 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 DDS middleware 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.

5.2 DDS_DataReaderListener Struct Reference

The DDS_DataReaderListener provides asynchronous notification of DDS_DataReader events.

Data Fields

- 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)

5.2.1 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.

5.2.2 Field Documentation

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

on_data_available() is called when the CoreDX DDS middleware 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.

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

on_liveliness_changed() is called when the CoreDX DDS middleware 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.

5.2.2.3 void(∗ DDS_DataReaderListener::on_requested_deadline_missed) (DDS_DataReader the_reader, DDS_RequestedDeadlineMissedStatus status)

on_requested_deadline_missed() is called when the CoreDX DDS middleware 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.

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

on_requested_incompatible_qos() is called when the CoreDX DDS middleware 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.

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

on_sample_lost() is called when the CoreDX DDS middleware 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.

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

on_sample_rejected() is called when the CoreDX DDS middleware detects that a sample has been rejected.

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

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

on_subscription_matched() is called when the CoreDX DDS middleware 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.

5.3 DDS_DataWriterListener_cd Struct Reference

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

Data Fields

- 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)

5.3.1 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.

5.3.2 Field Documentation

5.3.2.1 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 DDS 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.

5.3.2.2 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 DDS 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.

5.3.2.3 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 DDS 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.

5.3.2.4 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 DDS 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.

5.4 DDS_DataWriterListener Struct Reference

The DDS_DataWriterListener provides asynchronous notification of DDS_DataWriter events.

Data Fields

- 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)

5.4.1 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.

5.4.2 Field Documentation

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

on_liveliness_lost() is called when the CoreDX DDS 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.

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

on_offered_deadline_missed() is called when the CoreDX DDS 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.

5.4.2.3 void(∗ DDS_DataWriterListener::on_offered_incompatible_qos) (DDS_DataWriter writer, DDS_OfferedIncompatibleQosStatus status)

on_offered_incompatible_qos() is called when the CoreDX DDS 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.

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

on_publication_matched() is called when the CoreDX DDS 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.

5.5 DDS_DomainParticipantListener_cd Struct Reference

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

Data Fields

- 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)

5.5.1 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.

5.5.2 Field Documentation

5.5.2.1 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 DDS middleware 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.

5.5.2.2 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 DDS middleware 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.

5.5.2.3 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 DDS middleware 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.

5.5.2.4 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 DDS middleware 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.

5.5.2.5 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 DDS middleware 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.

5.5.2.6 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 DDS middleware 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.

5.5.2.7 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 DDS middleware 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.

5.5.2.8 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 DDS middleware 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.

5.5.2.9 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 DDS middleware 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.

5.5.2.10 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 DDS middleware 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.

5.5.2.11 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 DDS middleware 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.

5.5.2.12 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 DDS middleware 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.

5.5.2.13 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 DDS middleware 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.

5.6 DDS_DomainParticipantListener Struct Reference

The DDS_DomainParticipantListener provides asynchronous notification of DDS_DomainParticipant events.

Data Fields

- 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)

5.6.1 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.

5.6.2 Field Documentation

5.6.2.1 void(* DDS_DomainParticipantListener::on_data_available) (DDS_DataReader the_reader)

on_data_available() is called when the CoreDX DDS middleware 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.

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

on_data_on_readers() is called when the CoreDX DDS middleware 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.

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

on_inconsistent_topic() is called when the CoreDX DDS middleware 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.

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

on_liveliness_changed() is called when the CoreDX DDS middleware 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.

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

on_liveliness_lost() is called when the CoreDX DDS middleware 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.

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

on_offered_deadline_missed() is called when the CoreDX DDS middleware 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.

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

on_offered_incompatible_qos() is called when the CoreDX DDS middleware 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.

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

on_publication_matched() is called when the CoreDX DDS middleware 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.

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

on_requested_deadline_missed() is called when the CoreDX DDS middleware 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.

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

on_requested_incompatible_qos() is called when the CoreDX DDS middleware 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.

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

on_sample_lost() is called when the CoreDX DDS middleware 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.

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

on_sample_rejected() is called when the CoreDX DDS middleware 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.

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

on_subscription_matched() is called when the CoreDX DDS middleware 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.

5.7 DDS_PublisherListener_cd Struct Reference

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

Data Fields

- 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)

5.7.1 Detailed Description

The DDS_PublisherListener_cd provides asynchronous notification of DDS_Publisher events with additionall 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.

5.7.2 Field Documentation

5.7.2.1 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 DDS middleware 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.

5.7.2.2 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 DDS middleware 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.

5.7.2.3 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 DDS middleware 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.

5.7.2.4 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 DDS middleware 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.

5.8 DDS_PublisherListener Struct Reference

The DDS_PublisherListener provides asynchronous notification of DDS_Publisher events.

Data Fields

- 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)

5.8.1 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.

5.8.2 Field Documentation

5.8.2.1 void(∗ DDS_PublisherListener::on_liveliness_lost) (DDS_DataWriter writer, DDS_LivelinessLostStatus status)

on_liveliness_lost() is called when the CoreDX DDS 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.

5.8.2.2 void(∗ DDS_PublisherListener::on_offered_deadline_missed) (DDS_DataWriter writer, DDS_OfferedDeadlineMissedStatus status)

on_offered_deadline_missed() is called when the CoreDX DDS 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.

5.8.2.3 void(∗ DDS_PublisherListener::on_offered_incompatible_qos) (DDS_DataWriter writer, DDS_OfferedIncompatibleQosStatus status)

on_offered_incompatible_qos() is called when the CoreDX DDS 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.

5.8.2.4 void(* DDS_PublisherListener::on_publication_matched) (DDS_DataWriter writer, DDS_PublicationMatchedStatus status)

on_publication_matched() is called when the CoreDX DDS 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.

5.9 DDS_SubscriberListener_cd Struct Reference

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

Data Fields

- 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)

5.9.1 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.

5.9.2 Field Documentation

5.9.2.1 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 DDS middleware 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.

5.9.2.2 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 DDS middleware 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.

5.9.2.3 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 DDS middleware 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.

5.9.2.4 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 DDS middleware 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.

5.9.2.5 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 DDS middleware 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.

5.9.2.6 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 DDS middleware 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.

5.9.2.7 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 DDS middleware 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.

5.9.2.8 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 DDS middleware 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.

5.10 DDS_SubscriberListener Struct Reference

The DDS_SubscriberListener provides asynchronous notification of DDS_Subscriber events.

Data Fields

- 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)

5.10.1 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.

5.10.2 Field Documentation

5.10.2.1 void(* DDS_SubscriberListener::on_data_available) (DDS_DataReader the_reader)

on_data_available() is called when the CoreDX DDS middleware 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.

5.10.2.2 void(* DDS_SubscriberListener::on_data_on_readers) (DDS_Subscriber the_subscriber)

on_data_on_readers() is called when the CoreDX DDS middleware 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.

5.10.2.3 void(* DDS_SubscriberListener::on_liveliness_changed) (DDS_DataReader the_reader, DDS_LivelinessChangedStatus status)

on_liveliness_changed() is called when the CoreDX DDS middleware 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.

5.10.2.4 void(* DDS_SubscriberListener::on_requested_deadline_missed) (DDS_DataReader the_reader, DDS_RequestedDeadlineMissedStatus status)

on_requested_deadline_missed() is called when the CoreDX DDS middleware 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.

5.10.2.5 void(* DDS_SubscriberListener::on_requested_incompatible_qos) (DDS_DataReader the_reader, DDS_RequestedIncompatibleQosStatus status)

on_requested_incompatible_qos() is called when the CoreDX DDS middleware 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.

5.10.2.6 void(* DDS_SubscriberListener::on_sample_lost) (DDS_DataReader the_reader, DDS_SampleLostStatus status)

on_sample_lost() is called when the CoreDX DDS middleware 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.

5.10.2.7 void(* DDS_SubscriberListener::on_sample_rejected) (DDS_DataReader the_reader, DDS_SampleRejectedStatus status)

on_sample_rejected() is called when the CoreDX DDS middleware 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.

5.10.2.8 void(* DDS_SubscriberListener::on_subscription_matched) (DDS_DataReader the_reader, DDS_SubscriptionMatchedStatus status)

on_subscription_matched() is called when the CoreDX DDS middleware 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.

5.11 DDS_TopicListener_cd Struct Reference

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

Data Fields

- void(∗ on_inconsistent_topic)(struct DDS_TopicListener_cd ∗self, DDS_Topic the_topic, DDS_InconsistentTopicStatus status, void ∗callback_data)

5.11.1 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().

5.11.2 Field Documentation

5.11.2.1 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 DDS 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.

5.12 DDS_TopicListener Struct Reference

The DDS_TopicListener provides asynchronous notification of DDS_Topic events.

Data Fields

- void(∗ on_inconsistent_topic)(DDS_Topic the_topic, DDS_InconsistentTopicStatus status)

5.12.1 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.

5.12.2 Field Documentation

5.12.2.1 void(∗ DDS_TopicListener::on_inconsistent_topic) (DDS_Topic the_topic, DDS_InconsistentTopicStatus status)

on_inconsistent_topic() is called when the CoreDX DDS 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.

Chapter 6

DDS Status Structures

6.1 DDS_InconsistentTopicStatus Struct Reference

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

Data Fields

- 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.*
- char last_inconsistent_topic_msg [256]

  Description of the most recently seen inconsistency.

6.1.1 Detailed Description

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

6.2 DDS_LivelinessChangedStatus Struct Reference

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

Data Fields

- 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.

6.2.1 Detailed Description

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

6.3 DDS_LivelinessLostStatus Struct Reference

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

Data Fields

- 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.*

6.3.1 Detailed Description

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

6.4 DDS_OfferedDeadlineMissedStatus Struct Reference

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

Data Fields

- 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.

6.4.1 Detailed Description

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

6.5 DDS_OfferedIncompatibleQosStatus Struct Reference

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

Data Fields

- 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.

6.5.1 Detailed Description

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

6.6 DDS_PublicationMatchedStatus Struct Reference

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

Data Fields

- 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.*

6.6.1 Detailed Description

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

6.7 DDS_RequestedDeadlineMissedStatus Struct Reference

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

Data Fields

- 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.

6.7.1 Detailed Description

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

6.8 DDS_RequestedIncompatibleQosStatus Struct Reference

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

Data Fields

- 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.

6.8.1 Detailed Description

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

6.9 DDS_SampleLostStatus Struct Reference

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

Data Fields

- 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.*

6.9.1 Detailed Description

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

6.10 DDS_SampleRejectedStatus Struct Reference

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

Data Fields

- 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.

6.10.1 Detailed Description

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

6.11 DDS_SubscriptionMatchedStatus Struct Reference

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

Data Fields

- 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.*

6.11.1 Detailed Description

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

6.12 DDS_QosPolicyCount Struct Reference

Holds a DDS_QosPolicyId_t value and a counter to indicate the number of events associated with that PolicyId.

Data Fields

- DDS_QosPolicyId_t policy_id
- int count

6.12.1 Detailed Description

Holds a DDS_QosPolicyId_t value and a counter to indicate the number of events associated with that PolicyId.

6.12.2 Field Documentation

6.12.2.1 int DDS_QosPolicyCount::count

Counter associated with this PolicyId.

6.12.2.2 DDS_QosPolicyId_t DDS_QosPolicyCount::policy_id

QosPolicyCount value.

Chapter 7

Builtin DDS Types

7.1 DDS_BuiltinTopicKey_t Struct Reference

Data Fields

- int value [3]

7.1.1 Detailed Description

Holds the 'key' for a builtin topic

7.1.2 Field Documentation

7.1.2.1 int DDS_BuiltinTopicKey_t::value[3]

12 bytes of key data

7.2 DDS_Duration_t Struct Reference

The Duration_t structure contains data to define a time duration.

7.2.1 Detailed Description

The Duration_t structure contains data to define a time duration.

7.3 DDS_GUID_t Struct Reference

Data Fields

- unsigned char value [16]

7.3.1 Detailed Description

A complete GUID that uniquely identifies a DDS entity.

7.3.2 Field Documentation

7.3.2.1 unsigned char DDS_GUID_t::value[16]

the guid bytes

7.4 DDS_Property_t Struct Reference

A name-value pair property. The 'propagate' flag indicates if this property should be transfered through discovery.

Data Fields

- char * name
  the property name
- char * value
  the property value
- unsigned char propagate
  propagate over discovery?

7.4.1 Detailed Description

A name-value pair property. The 'propagate' flag indicates if this property should be transfered through discovery.

7.5 DDS_SampleIdentity_t Struct Reference

Data Fields

- struct DDS_GUID_t guid

7.5.1 Detailed Description

Uniquely identifies a sample and its source entity.

7.5.2 Field Documentation

7.5.2.1 struct DDS_GUID_t DDS_SampleIdentity_t::guid

The source of the sample. The sequence number of the sample

7.6 DDS_SequenceNumber_t Struct Reference

Data Fields

- int high
- uint32_t low

7.6.1 Detailed Description

A sequence number that uniquely identifies a sample within the scope of a writer cache.

7.6.2 Field Documentation

7.6.2.1 int DDS_SequenceNumber_t::high

The 'high' 32 bits of the sequence number

7.6.2.2 uint32_t DDS_SequenceNumber_t::low

The 'low' 32 bits of the sequence number

7.7 DDS_Time_t Struct Reference

Data Fields

- int sec
- uint32_t nanosec

7.7.1 Detailed Description

Defines a time

7.7.2 Field Documentation

7.7.2.1 uint32_t DDS_Time_t::nanosec

Time nanoseconds

7.7.2.2 int DDS_Time_t::sec

Time seconds

Chapter 8

DDS Discovery Types

8.1 DDS_DCPSParticipant Struct Reference

Data Fields

- struct DDS_BuiltinTopicKey_t key

  key for this participant (the 12 byte prefix)
- struct DDS_UserDataQosPolicy user_data

  user_data associated with this participant
- struct DDS_PropertyQosPolicy properties

  key-value pairs associated with this participant
- char * entity_name

  arbitrary string name assigned to participant

8.1.1 Detailed Description

The DCPSParticipant structure holds discovered Participant data.

8.2 DDS_DCPSPublication Struct Reference

Data Fields

- struct DDS_BuiltinTopicKey_t participant_key
  the prefix of containing participant
- struct DDS_BuiltinTopicKey_t key
  the entity id of this entity
- char * topic_name
  the topic name
- char * type_name
  type of the type
- struct DDS_DurabilityQosPolicy durability
  durability config of entity
- struct DDS_DurabilityServiceQosPolicy durability_service
  durability service config of this entity
- struct DDS_DeadlineQosPolicy deadline
  deadline config of this entity
- struct DDS_LatencyBudgetQosPolicy latency_budget
  latency budget config of this entity
- struct DDS_LivelinessQosPolicy liveliness
  liveliness config of this entity
- struct DDS_ReliabilityQosPolicy reliability
  reliability config of this entity
- struct DDS_LifespanQosPolicy lifespan
  lifespan config of this entity
- struct DDS_UserDataQosPolicy user_data
  user data config of this entity
- struct DDS_OwnershipQosPolicy ownership
  ownership config of this entity
- struct DDS_OwnershipStrengthQosPolicy ownership_strength
  ownership strength config of this entity
- struct DDS_DestinationOrderQosPolicy destination_order
  destination order config of this entity
- struct DDS_PresentationQosPolicy presentation
  presentation config of this entity
- struct DDS_PartitionQosPolicy partition
  partition config of this entity
- struct DDS_TopicDataQosPolicy topic_data
  topic data config of this entity
- struct DDS_GroupDataQosPolicy group_data
  group data config of this entity
- struct DDS_DataRepresentationQosPolicy representation
  data representation config of this entity

8.2.1 Detailed Description

The DCPSPublication structure holds discovered DataWriter data.

8.3 DDS_DCPSSubscription Struct Reference

Data Fields

- struct DDS_BuiltinTopicKey_t participant_key

  containing participant guid prefix config of this entity
- struct DDS_BuiltinTopicKey_t key

  the entity id this entity
- char * topic_name

  topic name config of this entity
- char * type_name

  type name config of this entity
- struct DDS_DurabilityQosPolicy durability

  durability config of this entity
- struct DDS_DeadlineQosPolicy deadline

  deadline config of this entity
- struct DDS_LatencyBudgetQosPolicy latency_budget

  latency budget config of this entity
- struct DDS_LivelinessQosPolicy liveliness

  liveliness config of this entity
- struct DDS_ReliabilityQosPolicy reliability

  realiability config of this entity
- struct DDS_OwnershipQosPolicy ownership

  ownership config of this entity
- struct DDS_DestinationOrderQosPolicy destination_order

  destination order config of this entity
- struct DDS_UserDataQosPolicy user_data

  user data config of this entity
- struct DDS_TimeBasedFilterQosPolicy time_based_filter

  time based filter config of this entity
- struct DDS_PresentationQosPolicy presentation

  presentation config of this entity
- struct DDS_PartitionQosPolicy partition

  partition config of this entity
- struct DDS_TopicDataQosPolicy topic_data

  topic data config of this entity
- struct DDS_GroupDataQosPolicy group_data

  group data config of this entity
- struct DDS_DataRepresentationQosPolicy representation

  representation config of this entity
- struct DDS_TypeConsistencyEnforcementQosPolicy type_consistency

  type consistency config of this entity
- char * entity_name

  entity name config of this entity
- struct DDS_RpcQosPolicy rpc

  rpc config of this entity

8.3.1 Detailed Description

The DCPSSubscription structure holds discovered DataReader data.

Chapter 9

Supporting Types

9.1 CoreDX_Locator Struct Reference

Network address.

Data Fields

- int kind
- uint32_t port
- unsigned char addr [16]

9.1.1 Detailed Description

Network address.

For a list of suppored LOCATOR_KINDs, refer to include/dds/dds.h

9.1.2 Field Documentation

9.1.2.1 unsigned char CoreDX_Locator::addr[16]

locator address

9.1.2.2 int CoreDX_Locator::kind

locator kind

9.1.2.3 uint32_t CoreDX_Locator::port

locator port

9.2 CoreDX_ParticipantLocator Struct Reference

Describes a the location and identity of a potential peer DomainParticipant.

Data Fields

- uint32_t participant_id

  participant id start
- uint32_t participant_id_max

  participant id max
- CoreDX_Locator participant_locator

  locator identifying peer participant

9.2.1 Detailed Description

Describes a the location and identity of a potential peer DomainParticipant.

9.3 DDS_CacheStatistics Struct Reference

Encapsulates statistics available from a DataReader or DataWriter.

Data Fields

- int32_t instance_count
- int32_t instance_alive_count
- int32_t instance_disposed_count
- int32_t instance_no_writers_count
- int32_t instance_new_count
- int32_t sample_count
- int32_t sample_read_count
- uint32_t state_flags

9.3.1 Detailed Description

Encapsulates statistics available from a DataReader or DataWriter.

Queries the current state of the sample and instance cache. A snapshot in time of the current state is written to the provided DDS_CacheStatistics parameter. The statistics include sample and instance counts.

Return values

DDS_RETCODE_OK	on success
DDS_RETCODE_NO_DATA	if unable to get the data

See also

DDS_DataReader_get_cache_stats()
DDS_DataWriter_get_cache_stats()

9.3.2 Field Documentation

9.3.2.1 int32_t DDS_CacheStatistics::instance_alive_count

number of alive instances

9.3.2.2 int32_t DDS_CacheStatistics::instance_count

total number of instances

9.3.2.3 int32_t DDS_CacheStatistics::instance_disposed_count

number of disposed instances

9.3.2.4 int32_t DDS_CacheStatistics::instance_new_count

number of 'new' (view state) instances

9.3.2.5 int32_t DDS_CacheStatistics::instance_no_writers_count

number of no_writer instances

9.3.2.6 int32_t DDS_CacheStatistics::sample_count

total number of samples

9.3.2.7 int32_t DDS_CacheStatistics::sample_read_count

number of 'read' samples

9.3.2.8 uint32_t DDS_CacheStatistics::state_flags

flags indicating internal state

Chapter 10

X-Types DynamicType API

10.1 DDS_AnnotationDescriptor Struct Reference

A DDS_AnnotationDescriptor object comprises the state of an annotation as it is applied to some element.

Data Fields

- DDS_DynamicType type
- DDS_Parameters parameters

Related Functions

(Note that these are not member functions.)

- DDS_ReturnCode_t DDS_AnnotationDescriptor_get_value (const DDS_AnnotationDescriptor *ad, DDS_ObjectName *value, const DDS_ObjectName key)

  This accesses an annotation property 'value' associated with the 'key' name.
- DDS_ReturnCode_t DDS_AnnotationDescriptor_get_all_value (const DDS_AnnotationDescriptor *ad, DDS_Parameters *value)

  Access the map of all key-value pairs.
- DDS_ReturnCode_t DDS_AnnotationDescriptor_set_value (DDS_AnnotationDescriptor *ad, const DDS_ObjectName key, const DDS_ObjectName value)
- DDS_ReturnCode_t DDS_AnnotationDescriptor_copy_from (DDS_AnnotationDescriptor *ad, const DDS_AnnotationDescriptor *other)
- unsigned char DDS_AnnotationDescriptor_equals (DDS_AnnotationDescriptor *ad, const DDS_AnnotationDescriptor *other)
- unsigned char DDS_AnnotationDescriptor_is_consistent (const DDS_AnnotationDescriptor *ad)
- DDS_DynamicType DDS_AnnotationDescriptor_get_type (const DDS_AnnotationDescriptor *ad)

10.1.1 Detailed Description

A DDS_AnnotationDescriptor object comprises the state of an annotation as it is applied to some element.

10.1.2 Field Documentation

10.1.2.1 DDS_Parameters DDS_AnnotationDescriptor::parameters

a map of key.value pairs

10.1.2.2 DDS_DynamicType DDS_AnnotationDescriptor::type

the annotation type

10.2 DDS_DynamicDataFactory Struct Reference

This type is responsible for creating DDS_DynamicData instances.

Related Functions

(Note that these are not member functions.)

- DDS_DynamicDataFactory DDS_DynamicDataFactory_get_instance (void)

  Return a DDS_DynamicDataFactory instance that can be used to create and delete DDS_DynamicData instances.
- DDS_ReturnCode_t DDS_DynamicDataFactory_delete_instance (void)

  Reclaim any resources associated with the object(s) previously returned from get_instance.
- DDS_DynamicData DDS_DynamicDataFactory_create_data (DDS_DynamicDataFactory ddf, DDS_DynamicType dt)

  Create and return a new data sample. All objects returned by this operation should eventually be deleted by calling DDS_DynamicDataFactory_delete_data.
- DDS_ReturnCode_t DDS_DynamicDataFactory_delete_data (DDS_DynamicDataFactory ddf, DDS_DynamicData data)

  Dispose of a data sample, reclaiming any associated resources.

10.2.1 Detailed Description

This type is responsible for creating DDS_DynamicData instances.

See also

DDS_DynamicType
DDS_DynamicData

10.3 DDS_DynamicDataReader Struct Reference

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

Related Functions

(Note that these are not member functions.)

- DDS_ReturnCode_t DDS_DynamicDataReader_read (DDS_DynamicDataReader dr, DDS_DynamicDataSeq *received_data, DDS_SampleInfoSeq *sample_infos, int max_samples, DDS_SampleStateMask sample_states, DDS_ViewStateMask view_states, DDS_InstanceStateMask instance_states)

  This operation accesses DDS_DynamicData data values (with associated DDS_SampleInfo) within the DataReader.
- DDS_ReturnCode_t DDS_DynamicDataReader_take (DDS_DynamicDataReader dr, DDS_DynamicDataSeq *received_data, DDS_SampleInfoSeq *sample_infos, int max_samples, DDS_SampleStateMask sample_states, DDS_ViewStateMask view_states, DDS_InstanceStateMask instance_states)

  This operation accesses DDS_DynamicData data values (with associated DDS_SampleInfo) within the DataReader.
- DDS_ReturnCode_t DDS_DynamicDataReader_read_w_condition (DDS_DynamicDataReader dr, DDS_DynamicDataSeq *received_data, DDS_SampleInfoSeq *sample_infos, int max_samples, DDS_ReadCondition a_condition)

  This operation accesses DDS_DynamicData data values (with associated DDS_SampleInfo) within the DataReader.
- DDS_ReturnCode_t DDS_DynamicDataReader_take_w_condition (DDS_DynamicDataReader dr, DDS_DynamicDataSeq *received_data, DDS_SampleInfoSeq *sample_infos, int max_samples, DDS_ReadCondition a_condition)

  This operation accesses DDS_DynamicData data values (with associated DDS_SampleInfo) within the DataReader.
- DDS_ReturnCode_t DDS_DynamicDataReader_read_next_sample (DDS_DynamicDataReader dr, DDS_DynamicData received_data, DDS_SampleInfo *sample_info)

  This operation accesses DDS_DynamicData data values (with associated DDS_SampleInfo) within the DataReader.
- DDS_ReturnCode_t DDS_DynamicDataReader_take_next_sample (DDS_DynamicDataReader dr, DDS_DynamicData received_data, DDS_SampleInfo *sample_info)

  This operation accesses DDS_DynamicData data values (with associated DDS_SampleInfo) within the DataReader.
- DDS_ReturnCode_t DDS_DynamicDataReader_read_instance (DDS_DynamicDataReader dr, DDS_DynamicDataSeq *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 DDS_DynamicData data values (with associated DDS_SampleInfo) within the DataReader.
- DDS_ReturnCode_t DDS_DynamicDataReader_take_instance (DDS_DynamicDataReader dr, DDS_DynamicDataSeq *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 DDS_DynamicData data values (with associated DDS_SampleInfo) within the DataReader.
- DDS_ReturnCode_t DDS_DynamicDataReader_read_next_instance (DDS_DynamicDataReader dr, DDS_DynamicDataSeq *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 DDS_DynamicData data values (with associated DDS_SampleInfo) within the DataReader.
- DDS_ReturnCode_t DDS_DynamicDataReader_take_next_instance (DDS_DynamicDataReader dr, DDS_DynamicDataSeq *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 DDS_DynamicData data values (with associated DDS_SampleInfo) within the DataReader.

- DDS_ReturnCode_t DDS_DynamicDataReader_read_next_instance_w_condition (DDS_DynamicDataReader dr, DDS_DynamicDataSeq *received_data, DDS_SampleInfoSeq *sample_infos, int max_samples, DDS_InstanceHandle_t previous_handle, DDS_ReadCondition a_condition)

  This operation accesses DDS_DynamicData data values (with associated DDS_SampleInfo) within the DataReader.
- DDS_ReturnCode_t DDS_DynamicDataReader_take_next_instance_w_condition (DDS_DynamicDataReader dr, DDS_DynamicDataSeq *received_data, DDS_SampleInfoSeq *sample_infos, int max_samples, DDS_InstanceHandle_t previous_handle, DDS_ReadCondition a_condition)

  This operation accesses DDS_DynamicData data values (with associated DDS_SampleInfo) within the DataReader.
- DDS_ReturnCode_t DDS_DynamicDataReader_return_loan (DDS_DynamicDataReader dr, DDS_DynamicDataSeq *received_data, DDS_SampleInfoSeq *sample_infos)

  This operation returns DDS_DynamicData data values to the DataReader.
- DDS_ReturnCode_t DDS_DynamicDataReader_get_key_value (DDS_DynamicDataReader dr, DDS_DynamicData 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_DynamicDataReader_lookup_instance (DDS_DynamicDataReader dr, DDS_DynamicData instance_data)

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

10.3.1 Detailed Description

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

See also

DDS_DataReader

10.4 DDS_DynamicData Struct Reference

A DDS_DynamicData object represents an individual data sample. It provides reflective getters and setters for the members of that sample.

Related Functions

(Note that these are not member functions.)

- DDS_ReturnCode_t DDS_DynamicData_get_type (DDS_DynamicData dd, DDS_DynamicType *dyn_type)

  This provides the type that defines the values within this sample.
- DDS_ReturnCode_t DDS_DynamicData_get_descriptor (DDS_DynamicData dd, DDS_MemberDescriptor *value, const DDS_MemberId id)

  This provides access to the descriptor for each value in this object, identified by the member ID.
- DDS_ReturnCode_t DDS_DynamicData_set_descriptor (DDS_DynamicData dd, const DDS_MemberId id, const DDS_MemberDescriptor *value)

  Set the descriptor of the member specified by 'id'.
- unsigned char DDS_DynamicData_equals (DDS_DynamicData dd, const DDS_DynamicData *other)

  Compare two DDS_DynamicData instances for equality.
- DDS_MemberId DDS_DynamicData_get_member_id_by_name (DDS_DynamicData dd, const DDS_ObjectName name)

  Access the 'id' of a member matching 'name'.
- DDS_MemberId DDS_DynamicData_get_member_id_at_index (DDS_DynamicData dd, const unsigned int index)

  Access the 'id' of a member at the specified index.
- unsigned int DDS_DynamicData_get_item_count (DDS_DynamicData dd)

  Access the "item count" of the DDS_DynamicData.
- DDS_ReturnCode_t DDS_DynamicData_clear_all_values (DDS_DynamicData dd)

  Clears all values from the instance.
- DDS_ReturnCode_t DDS_DynamicData_clear_nonkey_values (DDS_DynamicData dd)

  Clear all non-key values from the instance.
- DDS_ReturnCode_t DDS_DynamicData_clear_value (DDS_DynamicData dd, const DDS_MemberId id)

  Clear the specified value from the instance.
- DDS_DynamicData DDS_DynamicData_loan_value (DDS_DynamicData dd, const DDS_MemberId id)

  The "loan" operations loan to the application a DDS_DynamicData object representing a value within this sample.
- DDS_ReturnCode_t DDS_DynamicData_return_loaned_value (DDS_DynamicData dd, const DDS_DynamicData value)

  Return a "loaned" data sample.
- DDS_DynamicData DDS_DynamicData_clone (DDS_DynamicData dd)

  Create and return a new data sample with the same contents as this one.
- DDS_ReturnCode_t DDS_DynamicData_get_int32_value (DDS_DynamicData dd, int *value, const DDS_MemberId id)

  Access a 32bit integer value, identified by 'id', in the sample.
- DDS_ReturnCode_t DDS_DynamicData_set_int32_value (DDS_DynamicData dd, const DDS_MemberId id, const int value)

  Set a 32bit integer value, identified by 'id', in the sample.
- DDS_ReturnCode_t DDS_DynamicData_get_uint32_value (DDS_DynamicData dd, unsigned int *value, const DDS_MemberId id)

Access an unsigned 32bit integer value, identified by 'id', in the sample.

- DDS_ReturnCode_t DDS_DynamicData_set_uint32_value (DDS_DynamicData dd, const DDS_MemberId id, const unsigned int value)

  Set an unsigned 32bit integer value, identified by 'id', in the sample.
- DDS_ReturnCode_t DDS_DynamicData_get_int16_value (DDS_DynamicData dd, short *value, const DDS_MemberId id)

  Access a 16bit integer value, identified by 'id', in the sample.
- DDS_ReturnCode_t DDS_DynamicData_set_int16_value (DDS_DynamicData dd, const DDS_MemberId id, const short value)

  Set a 16bit integer value, identified by 'id', in the sample.
- DDS_ReturnCode_t DDS_DynamicData_get_uint16_value (DDS_DynamicData dd, unsigned short *value, const DDS_MemberId id)

  Access an unsigned 16bit integer value, identified by 'id', in the sample.
- DDS_ReturnCode_t DDS_DynamicData_set_uint16_value (DDS_DynamicData dd, const DDS_MemberId id, const unsigned short value)

  Set an unsigned 16bit integer value, identified by 'id', in the sample.
- DDS_ReturnCode_t DDS_DynamicData_get_int64_value (DDS_DynamicData dd, int64_t *value, const DDS_MemberId id)

  Access a 64bit integer value, identified by 'id', in the sample.
- DDS_ReturnCode_t DDS_DynamicData_set_int64_value (DDS_DynamicData dd, const DDS_MemberId id, const int64_t value)

  Set a 64bit integer value, identified by 'id', in the sample.
- DDS_ReturnCode_t DDS_DynamicData_get_uint64_value (DDS_DynamicData dd, uint64_t *value, const DDS_MemberId id)

  Access an unsigned 64bit integer value, identified by 'id', in the sample.
- DDS_ReturnCode_t DDS_DynamicData_set_uint64_value (DDS_DynamicData dd, const DDS_MemberId id, const uint64_t value)

  Set an unsigned 64bit integer value, identified by 'id', in the sample.
- DDS_ReturnCode_t DDS_DynamicData_get_float32_value (DDS_DynamicData dd, float *value, const DDS_MemberId id)

  Access a float value, identified by 'id', in the sample.
- DDS_ReturnCode_t DDS_DynamicData_set_float32_value (DDS_DynamicData dd, const DDS_MemberId id, const float value)

  Set a float value, identified by 'id', in the sample.
- DDS_ReturnCode_t DDS_DynamicData_get_float64_value (DDS_DynamicData dd, double *value, const DDS_MemberId id)

  Access a double value, identified by 'id', in the sample.
- DDS_ReturnCode_t DDS_DynamicData_set_float64_value (DDS_DynamicData dd, const DDS_MemberId id, const double value)

  Set a double value, identified by 'id', in the sample.
- DDS_ReturnCode_t DDS_DynamicData_get_char8_value (DDS_DynamicData dd, char *value, const DDS_MemberId id)

  Access an 8bit character value, identified by 'id', in the sample.
- DDS_ReturnCode_t DDS_DynamicData_set_char8_value (DDS_DynamicData dd, const DDS_MemberId id, const char value)

  Set an 8bit character value, identified by 'id', in the sample.
- DDS_ReturnCode_t DDS_DynamicData_get_char32_value (DDS_DynamicData dd, cdx_char32_t *value, const DDS_MemberId id)

  Access a 32bit character value, identified by 'id', in the sample.

- DDS_ReturnCode_t DDS_DynamicData_set_char32_value (DDS_DynamicData dd, const DDS_MemberId id, const cdx_char32_t value)

  Set a 32bit character value, identified by 'id', in the sample.
- DDS_ReturnCode_t DDS_DynamicData_get_byte_value (DDS_DynamicData dd, unsigned char *value, const DDS_MemberId id)

  Access an 8bit value, identified by 'id', in the sample.
- DDS_ReturnCode_t DDS_DynamicData_set_byte_value (DDS_DynamicData dd, const DDS_MemberId id, const unsigned char value)

  Set an 8bit value, identified by 'id', in the sample.
- DDS_ReturnCode_t DDS_DynamicData_get_boolean_value (DDS_DynamicData dd, unsigned char *value, const DDS_MemberId id)

  Access a boolean value, identified by 'id', in the sample.
- DDS_ReturnCode_t DDS_DynamicData_set_boolean_value (DDS_DynamicData dd, const DDS_MemberId id, const unsigned char value)

  Set a boolean value, identified by 'id', in the sample.
- DDS_ReturnCode_t DDS_DynamicData_get_string_value (DDS_DynamicData dd, char **value, const DDS_MemberId id)

  Access a string value, identified by 'id', in the sample.
- DDS_ReturnCode_t DDS_DynamicData_set_string_value (DDS_DynamicData dd, const DDS_MemberId id, const char *value)

  Set a string value, identified by 'id', in the sample.
- DDS_ReturnCode_t DDS_DynamicData_get_wstring_value (DDS_DynamicData dd, cdx_char32_t **value, const DDS_MemberId id)

  Access a wstring value, identified by 'id', in the sample.
- DDS_ReturnCode_t DDS_DynamicData_set_wstring_value (DDS_DynamicData dd, const DDS_MemberId id, const cdx_char32_t *value)

  Set a wstring value, identified by 'id', in the sample.
- DDS_ReturnCode_t DDS_DynamicData_get_complex_value (DDS_DynamicData dd, DDS_DynamicData *value, const DDS_MemberId id)

  Access a complex value, identified by 'id', in the sample.
- DDS_ReturnCode_t DDS_DynamicData_set_complex_value (DDS_DynamicData dd, const DDS_MemberId id, const DDS_DynamicData value)

  Set a complex value, identified by 'id', in the sample.
- DDS_ReturnCode_t DDS_DynamicData_get_map_key_value (DDS_DynamicData dd, const char **key_str, const DDS_MemberId id)

  Access the key value of a map type, identified by 'id', in the sample.
- DDS_ReturnCode_t DDS_DynamicData_get_int32_values (DDS_DynamicData dd, DDS_Int32Seq *value, const DDS_MemberId id)

  Provide access to array values as a sequence. If 'dd' is an array type, then 'id' must be DDS_MEMBER_ID_INVALID. If 'dd' is an aggretage type, then 'id' must refer to a member that is of type array. The sequence is populated with a 'shallow' copy of the data. This means that modifications to any of the values in the sequence will result in modifications to the data in the DDS_DynamicData instance. It is an error to 'clear' the returned sequence, as this will destroy memory that the DynamicData owns.
- DDS_ReturnCode_t DDS_DynamicData_set_int32_values (DDS_DynamicData dd, const DDS_MemberId id, const DDS_Int32Seq *value)

  Modify an array of values in the DDS_DynamicData instance.
- DDS_ReturnCode_t DDS_DynamicData_get_uint32_values (DDS_DynamicData dd, DDS_UInt32Seq *value, const DDS_MemberId id)

Provide access to array values as a sequence. If 'dd' is an array type, then 'id' must be DDS_MEMBER_ID_INVALID. If 'dd' is an aggretage type, then 'id' must refer to a member that is of type array. The sequence is populated with a 'shallow' copy of the data. This means that modifications to any of the values in the sequence will result in modifications to the data in the DDS_DynamicData instance. It is an error to 'clear' the returned sequence, as this will destroy memory that the DynamicData owns.

- DDS_ReturnCode_t DDS_DynamicData_set_uint32_values (DDS_DynamicData dd, const DDS_MemberId id, const DDS_UInt32Seq ∗value)

  Modify an array of values in the DDS_DynamicData instance. If 'dd' is an array type, then 'id' must be DDS_MEMBER_ID_INVALID. If 'dd' is an aggretage type, then 'id' must refer to a member that is of type array. This copies the data into the DynamicData instance 'dd'. Modifications to the value sequence after this call will not have an effect on the contents of 'dd'.

- DDS_ReturnCode_t DDS_DynamicData_get_int16_values (DDS_DynamicData dd, DDS_Int16Seq ∗value, const DDS_MemberId id)

  Provide access to array values as a sequence. If 'dd' is an array type, then 'id' must be DDS_MEMBER_ID_INVALID. If 'dd' is an aggretage type, then 'id' must refer to a member that is of type array. The sequence is populated with a 'shallow' copy of the data. This means that modifications to any of the values in the sequence will result in modifications to the data in the DDS_DynamicData instance. It is an error to 'clear' the returned sequence, as this will destroy memory that the DynamicData owns.

- DDS_ReturnCode_t DDS_DynamicData_set_int16_values (DDS_DynamicData dd, const DDS_MemberId id, const DDS_Int16Seq ∗value)

  Modify an array of values in the DDS_DynamicData instance. If 'dd' is an array type, then 'id' must be DDS_MEMBER_ID_INVALID. If 'dd' is an aggretage type, then 'id' must refer to a member that is of type array. This copies the data into the DynamicData instance 'dd'. Modifications to the value sequence after this call will not have an effect on the contents of 'dd'.

- DDS_ReturnCode_t DDS_DynamicData_get_uint16_values (DDS_DynamicData dd, DDS_UInt16Seq ∗value, const DDS_MemberId id)

  Provide access to array values as a sequence. If 'dd' is an array type, then 'id' must be DDS_MEMBER_ID_INVALID. If 'dd' is an aggretage type, then 'id' must refer to a member that is of type array. The sequence is populated with a 'shallow' copy of the data. This means that modifications to any of the values in the sequence will result in modifications to the data in the DDS_DynamicData instance. It is an error to 'clear' the returned sequence, as this will destroy memory that the DynamicData owns.

- DDS_ReturnCode_t DDS_DynamicData_set_uint16_values (DDS_DynamicData dd, const DDS_MemberId id, const DDS_UInt16Seq ∗value)

  Modify an array of values in the DDS_DynamicData instance. If 'dd' is an array type, then 'id' must be DDS_MEMBER_ID_INVALID. If 'dd' is an aggretage type, then 'id' must refer to a member that is of type array. This copies the data into the DynamicData instance 'dd'. Modifications to the value sequence after this call will not have an effect on the contents of 'dd'.

- DDS_ReturnCode_t DDS_DynamicData_get_int64_values (DDS_DynamicData dd, DDS_Int64Seq ∗value, const DDS_MemberId id)

  Provide access to array values as a sequence. If 'dd' is an array type, then 'id' must be DDS_MEMBER_ID_INVALID. If 'dd' is an aggretage type, then 'id' must refer to a member that is of type array. The sequence is populated with a 'shallow' copy of the data. This means that modifications to any of the values in the sequence will result in modifications to the data in the DDS_DynamicData instance. It is an error to 'clear' the returned sequence, as this will destroy memory that the DynamicData owns.

- DDS_ReturnCode_t DDS_DynamicData_set_int64_values (DDS_DynamicData dd, const DDS_MemberId id, const DDS_Int64Seq ∗value)

  Modify an array of values in the DDS_DynamicData instance. If 'dd' is an array type, then 'id' must be DDS_MEMBER_ID_INVALID. If 'dd' is an aggretage type, then 'id' must refer to a member that is of type array. This copies the data into the DynamicData instance 'dd'. Modifications to the value sequence after this call will not have an effect on the contents of 'dd'.

- DDS_ReturnCode_t DDS_DynamicData_get_uint64_values (DDS_DynamicData dd, DDS_UInt64Seq ∗value, const DDS_MemberId id)

  Provide access to array values as a sequence. If 'dd' is an array type, then 'id' must be DDS_MEMBER_ID_INVALID. If 'dd' is an aggretage type, then 'id' must refer to a member that is of type array. The sequence is populated with a 'shallow'

copy of the data. This means that modifications to any of the values in the sequence will result in modifications to the data in the DDS_DynamicData instance. It is an error to 'clear' the returned sequence, as this will destroy memory that the DynamicData owns.

- DDS_ReturnCode_t DDS_DynamicData_set_uint64_values (DDS_DynamicData dd, const DDS_MemberId id, const DDS_UInt64Seq *value)

  Modify an array of values in the DDS_DynamicData instance. If 'dd' is an array type, then 'id' must be DDS_MEMBER_ID_INVALID. If 'dd' is an aggretage type, then 'id' must refer to a member that is of type array. This copies the data into the DynamicData instance 'dd'. Modifications to the value sequence after this call will not have an effect on the contents of 'dd'.

- DDS_ReturnCode_t DDS_DynamicData_get_float32_values (DDS_DynamicData dd, DDS_Float32Seq *value, const DDS_MemberId id)

  Provide access to array values as a sequence. If 'dd' is an array type, then 'id' must be DDS_MEMBER_ID_INVALID. If 'dd' is an aggretage type, then 'id' must refer to a member that is of type array. The sequence is populated with a 'shallow' copy of the data. This means that modifications to any of the values in the sequence will result in modifications to the data in the DDS_DynamicData instance. It is an error to 'clear' the returned sequence, as this will destroy memory that the DynamicData owns.

- DDS_ReturnCode_t DDS_DynamicData_set_float32_values (DDS_DynamicData dd, const DDS_MemberId id, const DDS_Float32Seq *value)

  Modify an array of values in the DDS_DynamicData instance. If 'dd' is an array type, then 'id' must be DDS_MEMBER_ID_INVALID. If 'dd' is an aggretage type, then 'id' must refer to a member that is of type array. This copies the data into the DynamicData instance 'dd'. Modifications to the value sequence after this call will not have an effect on the contents of 'dd'.

- DDS_ReturnCode_t DDS_DynamicData_get_float64_values (DDS_DynamicData dd, DDS_Float64Seq *value, const DDS_MemberId id)

  Provide access to array values as a sequence. If 'dd' is an array type, then 'id' must be DDS_MEMBER_ID_INVALID. If 'dd' is an aggretage type, then 'id' must refer to a member that is of type array. The sequence is populated with a 'shallow' copy of the data. This means that modifications to any of the values in the sequence will result in modifications to the data in the DDS_DynamicData instance. It is an error to 'clear' the returned sequence, as this will destroy memory that the DynamicData owns.

- DDS_ReturnCode_t DDS_DynamicData_set_float64_values (DDS_DynamicData dd, const DDS_MemberId id, const DDS_Float64Seq *value)

  Modify an array of values in the DDS_DynamicData instance. If 'dd' is an array type, then 'id' must be DDS_MEMBER_ID_INVALID. If 'dd' is an aggretage type, then 'id' must refer to a member that is of type array. This copies the data into the DynamicData instance 'dd'. Modifications to the value sequence after this call will not have an effect on the contents of 'dd'.

- DDS_ReturnCode_t DDS_DynamicData_get_char8_values (DDS_DynamicData dd, DDS_CharSeq *value, const DDS_MemberId id)

  Provide access to array values as a sequence. If 'dd' is an array type, then 'id' must be DDS_MEMBER_ID_INVALID. If 'dd' is an aggretage type, then 'id' must refer to a member that is of type array. The sequence is populated with a 'shallow' copy of the data. This means that modifications to any of the values in the sequence will result in modifications to the data in the DDS_DynamicData instance. It is an error to 'clear' the returned sequence, as this will destroy memory that the DynamicData owns.

- DDS_ReturnCode_t DDS_DynamicData_set_char8_values (DDS_DynamicData dd, const DDS_MemberId id, const DDS_CharSeq *value)

  Modify an array of values in the DDS_DynamicData instance. If 'dd' is an array type, then 'id' must be DDS_MEMBER_ID_INVALID. If 'dd' is an aggretage type, then 'id' must refer to a member that is of type array. This copies the data into the DynamicData instance 'dd'. Modifications to the value sequence after this call will not have an effect on the contents of 'dd'.

- DDS_ReturnCode_t DDS_DynamicData_get_char32_values (DDS_DynamicData dd, DDS_WcharSeq *value, const DDS_MemberId id)

  Provide access to array values as a sequence. If 'dd' is an array type, then 'id' must be DDS_MEMBER_ID_INVALID. If 'dd' is an aggretage type, then 'id' must refer to a member that is of type array. The sequence is populated with a 'shallow' copy of the data. This means that modifications to any of the values in the sequence will result in modifications to the data in the DDS_DynamicData instance. It is an error to 'clear' the returned sequence, as this will destroy memory that the DynamicData owns.

- DDS_ReturnCode_t DDS_DynamicData_set_char32_values (DDS_DynamicData dd, const DDS_MemberId id, const DDS_WcharSeq ∗value)

  Modify an array of values in the DDS_DynamicData instance. If 'dd' is an array type, then 'id' must be DDS_MEMBER_ID_INVALID. If 'dd' is an aggretage type, then 'id' must refer to a member that is of type array. This copies the data into the DynamicData instance 'dd'. Modifications to the value sequence after this call will not have an effect on the contents of 'dd'.
- DDS_ReturnCode_t DDS_DynamicData_get_byte_values (DDS_DynamicData dd, DDS_ByteSeq ∗value, const DDS_MemberId id)

  Provide access to array values as a sequence. If 'dd' is an array type, then 'id' must be DDS_MEMBER_ID_INVALID. If 'dd' is an aggretage type, then 'id' must refer to a member that is of type array. The sequence is populated with a 'shallow' copy of the data. This means that modifications to any of the values in the sequence will result in modifications to the data in the DDS_DynamicData instance. It is an error to 'clear' the returned sequence, as this will destroy memory that the DynamicData owns.
- DDS_ReturnCode_t DDS_DynamicData_set_byte_values (DDS_DynamicData dd, const DDS_MemberId id, const DDS_ByteSeq ∗value)

  Modify an array of values in the DDS_DynamicData instance. If 'dd' is an array type, then 'id' must be DDS_MEMBER_ID_INVALID. If 'dd' is an aggretage type, then 'id' must refer to a member that is of type array. This copies the data into the DynamicData instance 'dd'. Modifications to the value sequence after this call will not have an effect on the contents of 'dd'.
- DDS_ReturnCode_t DDS_DynamicData_get_boolean_values (DDS_DynamicData dd, DDS_BooleanSeq ∗value, const DDS_MemberId id)

  Provide access to array values as a sequence. If 'dd' is an array type, then 'id' must be DDS_MEMBER_ID_INVALID. If 'dd' is an aggretage type, then 'id' must refer to a member that is of type array. The sequence is populated with a 'shallow' copy of the data. This means that modifications to any of the values in the sequence will result in modifications to the data in the DDS_DynamicData instance. It is an error to 'clear' the returned sequence, as this will destroy memory that the DynamicData owns.
- DDS_ReturnCode_t DDS_DynamicData_set_boolean_values (DDS_DynamicData dd, const DDS_MemberId id, const DDS_BooleanSeq ∗value)

  Modify an array of values in the DDS_DynamicData instance. If 'dd' is an array type, then 'id' must be DDS_MEMBER_ID_INVALID. If 'dd' is an aggretage type, then 'id' must refer to a member that is of type array. This copies the data into the DynamicData instance 'dd'. Modifications to the value sequence after this call will not have an effect on the contents of 'dd'.
- DDS_ReturnCode_t DDS_DynamicData_get_string_values (DDS_DynamicData dd, DDS_StringSeq ∗value, const DDS_MemberId id)

  Provide access to array values as a sequence. If 'dd' is an array type, then 'id' must be DDS_MEMBER_ID_INVALID. If 'dd' is an aggretage type, then 'id' must refer to a member that is of type array. The sequence is populated with a 'shallow' copy of the data. This means that modifications to any of the values in the sequence will result in modifications to the data in the DDS_DynamicData instance. It is an error to 'clear' the returned sequence, as this will destroy memory that the DynamicData owns.
- DDS_ReturnCode_t DDS_DynamicData_set_string_values (DDS_DynamicData dd, const DDS_MemberId id, const DDS_StringSeq ∗value)

  Modify an array of values in the DDS_DynamicData instance. If 'dd' is an array type, then 'id' must be DDS_MEMBER_ID_INVALID. If 'dd' is an aggretage type, then 'id' must refer to a member that is of type array. This copies the data into the DynamicData instance 'dd'. Modifications to the value sequence after this call will not have an effect on the contents of 'dd'.
- DDS_ReturnCode_t DDS_DynamicData_get_wstring_values (DDS_DynamicData dd, DDS_WstringSeq ∗value, const DDS_MemberId id)

  Provide access to array values as a sequence. If 'dd' is an array type, then 'id' must be DDS_MEMBER_ID_INVALID. If 'dd' is an aggretage type, then 'id' must refer to a member that is of type array. The sequence is populated with a 'shallow' copy of the data. This means that modifications to any of the values in the sequence will result in modifications to the data in the DDS_DynamicData instance. It is an error to 'clear' the returned sequence, as this will destroy memory that the DynamicData owns.
- DDS_ReturnCode_t DDS_DynamicData_set_wstring_values (DDS_DynamicData dd, const DDS_MemberId id, const DDS_WstringSeq ∗value)

Modify an array of values in the DDS_DynamicData instance. If 'dd' is an array type, then 'id' must be DDS_MEMBER_ID_INVALID. If 'dd' is an aggretage type, then 'id' must refer to a member that is of type array. This copies the data into the DynamicData instance 'dd'. Modifications to the value sequence after this call will not have an effect on the contents of 'dd'.

- DDS_ReturnCode_t DDS_DynamicDataFactory_create_value (DDS_DynamicDataFactory ddf, const DDS_DynamicData data, const DDS_MemberId id, DDS_DynamicData ∗value)

  Creates a DynamicData instance for the specified member (id) of 'data'. This is useful for 'optional' members that are not constructed by the DDS_DynamicData_create_data() call. By default 'optional' members are left in the 'null' state. To set them, call create_value() and then set_complex_value().

10.4.1 Detailed Description

A DDS_DynamicData object represents an individual data sample. It provides reflective getters and setters for the members of that sample.

Many of the properties and operations on DDS_DynamicData refer to values within the sample, which are identified by name, member ID, or index.

What constitutes a value within a sample, and which means of accessing it are valid, depends on the type of the sample.

- If this instance is of an aggregated type, values correspond to the type's members and can be accessed by name, member ID, or index.
- If this instance is of a sequence or string type, values correspond to the elements of the collection. These elements must be accessed by index; the mapping from index to member ID is unspecified.
- If this object is of a map type, values correspond to the values of the map. Map keys are implicitly converted to strings and can thus be used to look up map values by name. Map values can also be accessed by index, although the order is unspecified.
- If the object is of an array type, values correspond to the elements of the array. These elements must be accessed by index; the mapping from index to member ID is unspecified. If the array is multi-dimensional, elements are accessed as if they were "flattened" into a single-dimensional array in the order specified by the IDL specification.
- If the object is of a bit set type, values correspond to the flags within the bit set and are all of Boolean type. Named flags can be accessed using that name; any bit within the bound of the bit set may be accessed by its index. The mappings from name and index to member ID are unspecified.
- If the object is of an enumeration or primitive type, it has no contained values. However, the value of the sample itself may be indicated by "name" using a nil or empty string, by "ID" by passing MEMBER_ID_INVALID, or by "index" by passing index 0.

Note that indices used here are always relative to other values in a particular DDS_DynamicData object. Even though member definitions within aggregate types have a well-defined order, the same is not true within data samples or across data samples.

Specifically, the index at which a member of an aggregated type appears in a particular data sample may not match that in which it appears in the corresponding type and may not match the index at which it appears in a different data sample.

There are several reasons for these inconsistencies:

- The producer of the sample may be using a slightly different variant of the type than the consumer, which may add to, or omit elements from, the set of members known to the consumer.
- An optional member may have no value; in such a case, it will be omitted, thereby decreasing the index of every subsequent member.

- A non-optional member may likewise be omitted (which semantically is equivalent to it taking its default value). An implementation may discretionarily omit such members (e.g., to save space).
- Preserving member order is not necessary or even desirable (e.g., for performance reasons) for certain data representations.

See also

DDS_DynamicType
DDS_DynamicDataFactory

10.5 DDS_DynamicDataWriter Struct Reference

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

Related Functions

(Note that these are not member functions.)

- DDS_InstanceHandle_t DDS_DynamicDataWriter_register_instance (DDS_DynamicDataWriter dw, const DDS_DynamicData instance_data)

  This operation registers a DDS_DynamicData data value.
- DDS_InstanceHandle_t DDS_DynamicDataWriter_register_instance_w_timestamp (DDS_DynamicDataWriter dw, const DDS_DynamicData instance_data, const DDS_Time_t source_timestamp)

  This operation registers a DDS_DynamicData data value.
- DDS_ReturnCode_t DDS_DynamicDataWriter_unregister_instance (DDS_DynamicDataWriter dw, const DDS_DynamicData instance_data, const DDS_InstanceHandle_t handle)

  This operation unregisters a DDS_DynamicData data value.
- DDS_ReturnCode_t DDS_DynamicDataWriter_unregister_instance_w_timestamp (DDS_DynamicDataWriter dw, const DDS_DynamicData instance_data, const DDS_InstanceHandle_t handle, const DDS_Time_t source_timestamp)

  This operation unregisters a DDS_DynamicData data value.
- DDS_ReturnCode_t DDS_DynamicDataWriter_write (DDS_DynamicDataWriter dw, const DDS_DynamicData instance_data, const DDS_InstanceHandle_t handle)

  This operation publishes a DDS_DynamicData data value.
- DDS_ReturnCode_t DDS_DynamicDataWriter_write_w_timestamp (DDS_DynamicDataWriter dw, const DDS_DynamicData instance_data, const DDS_InstanceHandle_t handle, const DDS_Time_t source_timestamp)

  This operation publishes a DDS_DynamicData data value.
- DDS_ReturnCode_t DDS_DynamicDataWriter_dispose (DDS_DynamicDataWriter dw, const DDS_DynamicData instance_data, const DDS_InstanceHandle_t instance_handle)

  This operation disposes a DDS_DynamicData data value.
- DDS_ReturnCode_t DDS_DynamicDataWriter_dispose_w_timestamp (DDS_DynamicDataWriter dw, const DDS_DynamicData instance_data, const DDS_InstanceHandle_t instance_handle, const DDS_Time_t source_timestamp)

  This operation disposes a DDS_DynamicData data value.
- DDS_ReturnCode_t DDS_DynamicDataWriter_get_key_value (DDS_DynamicDataWriter dw, DDS_DynamicData 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_DynamicDataWriter_lookup_instance (DDS_DynamicDataWriter dw, const DDS_DynamicData instance_data)

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

10.5.1 Detailed Description

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

See also

DDS_DataWriter

10.6 DDS_DynamicTypeBuilderFactory Struct Reference

An instance of this type is responsible for creating DDS_DynamicType and DDS_DynamicTypeSupport objects.

Related Functions

(Note that these are not member functions.)

- DDS_DynamicTypeBuilderFactory DDS_DynamicTypeBuilderFactory_get_instance (void)

  Return the DDS_DynamicTypeBuilderFactory singleton instance.
- DDS_ReturnCode_t DDS_DynamicTypeBuilderFactory_delete_instance (void)

  Reclaim any resources associated with the instance returned from DDS_DynamicTypeBuilderFactory_get_instance.
- DDS_DynamicType DDS_DynamicTypeBuilderFactory_get_primitive_type (DDS_DynamicTypeBuilderFactory dtbf, const DDS_TypeKind kind)

  Retrieve a DDS_DynamicType object corresponding to the indicated primitive type kind.
- DDS_DynamicTypeBuilder DDS_DynamicTypeBuilderFactory_create_type (DDS_DynamicTypeBuilderFactory dtbf, const DDS_TypeDescriptor *descriptor)

  Create and return a new DDS_DynamicTypeBuilder object as described by the given type descriptor.
- DDS_DynamicTypeBuilder DDS_DynamicTypeBuilderFactory_create_type_copy (DDS_DynamicTypeBuilderFactory dtbf, const DDS_DynamicType type)

  Create and return a new DDS_DynamicTypeBuilder object with a copy of the state of the given type.
- DDS_DynamicTypeBuilder DDS_DynamicTypeBuilderFactory_create_type_w_type_object (DDS_DynamicTypeBuilderFactory dtbf, const DDS_TypeObject *type_object)

  Create and return a new DDS_DynamicTypeBuilder object that describes a type identical to that described by the given TypeObject.
- DDS_DynamicTypeBuilder DDS_DynamicTypeBuilderFactory_create_string_type (DDS_DynamicTypeBuilderFactory dtbf, const unsigned int bound)

  Create and return a new DDS_DynamicTypeBuilder object representing a string type.
- DDS_DynamicTypeBuilder DDS_DynamicTypeBuilderFactory_create_wstring_type (DDS_DynamicTypeBuilderFactory dtbf, const unsigned int bound)

  Create and return a new DDS_DynamicTypeBuilder object representing a string type.
- DDS_DynamicTypeBuilder DDS_DynamicTypeBuilderFactory_create_sequence_type (DDS_DynamicTypeBuilderFactory dtbf, const DDS_DynamicType element_type, const unsigned int bound)

  Create and return a new DDS_DynamicTypeBuilder object representing a sequence type.
- DDS_DynamicTypeBuilder DDS_DynamicTypeBuilderFactory_create_array_type (DDS_DynamicTypeBuilderFactory dtbf, const DDS_DynamicType element_type, const DDS_BoundSeq *bound)

  Create and return a new DDS_DynamicTypeBuilder object representing an array type.
- DDS_DynamicTypeBuilder DDS_DynamicTypeBuilderFactory_create_map_type (DDS_DynamicTypeBuilderFactory dtbf, const DDS_DynamicType key_element_type, const DDS_DynamicType element_type, const unsigned int bound)

  Create and return a new DDS_DynamicTypeBuilder object representing a map type.
- DDS_DynamicTypeBuilder DDS_DynamicTypeBuilderFactory_create_bitset_type (DDS_DynamicTypeBuilderFactory dtbf, const unsigned int bound)

  Create and return a new DDS_DynamicTypeBuilder object representing a bit set type.
- DDS_DynamicTypeBuilder DDS_DynamicTypeBuilderFactory_create_type_w_uri (DDS_DynamicTypeBuilderFactory dtbf, const char *document_url, const char *type_name, const DDS_IncludePathSeq *include_paths)

  Create and return a new DDS_DynamicTypeBuilder object by parsing the type description at the given URL.
- DDS_DynamicTypeBuilder DDS_DynamicTypeBuilderFactory_create_type_w_document (DDS_DynamicTypeBuilderFactory dtbf, const char *document, const char *type_name, const DDS_IncludePathSeq *include_paths)

Create and return a new DDS_DynamicTypeBuilder object by parsing the type description contained in the given string.

- DDS_ReturnCode_t DDS_DynamicTypeBuilderFactory_delete_type_builder (DDS_DynamicTypeBuilderFactory dtbf, DDS_DynamicTypeBuilder type_builder)

  Destroy a DDS_DynamicTypeBuilder instance obtained through one of the create methods on the DDS_DynamicTypeBuilderFactory.
- DDS_ReturnCode_t DDS_DynamicTypeBuilderFactory_delete_type (DDS_DynamicTypeBuilderFactory dtbf, DDS_DynamicType type)

  Destroy a DDS_DynamicType instance obtained through the DDS_DynamicTypeBuilderFactory_get_primitive_type operation.
- DDS_DynamicTypeBuilder DDS_DynamicTypeBuilderFactory_create_structure_type (DDS_DynamicTypeBuilderFactory dtbf, DDS_DynamicType base_type)

  Create and return a new DDS_DynamicTypeBuilder object representing a structure type.
- DDS_DynamicTypeBuilder DDS_DynamicTypeBuilderFactory_create_union_type (DDS_DynamicTypeBuilderFactory dtbf)

  Create and return a new DDS_DynamicTypeBuilder object representing a union type.
- DDS_DynamicTypeBuilder DDS_DynamicTypeBuilderFactory_create_alias_type (DDS_DynamicTypeBuilderFactory dtbf, const DDS_DynamicType base_type)

  Create and return a new DDS_DynamicTypeBuilder object representing an alias for the provided 'base_type'.
- DDS_DynamicTypeBuilder DDS_DynamicTypeBuilderFactory_create_enumeration_type (DDS_DynamicTypeBuilderFactory dtbf, const unsigned int bound)

  Create and return a new DDS_DynamicTypeBuilder object representing an enumeration type.
- DDS_ReturnCode_t DDS_DynamicTypeBuilderFactory_create_extensibility_annotation (DDS_DynamicTypeBuilderFactory dtbf, DDS_AnnotationDescriptor *ad, DDS_ExtensibilityKind ekind)

  Initialize an AnnotationDescriptor to represent an '@extensibility' annotation with the specified extensibility kind.
- DDS_ReturnCode_t DDS_DynamicTypeBuilderFactory_create_optional_annotation (DDS_DynamicTypeBuilderFactory dtbf, DDS_AnnotationDescriptor *ad)

  Initialize an AnnotationDescriptor to represent an '@optional' annotation.

10.6.1 Detailed Description

An instance of this type is responsible for creating DDS_DynamicType and DDS_DynamicTypeSupport objects.

See also

DDS_DynamicType
DDS_DynamicTypeBuilder
DDS_DynamicTypeSupport

10.7 DDS_DynamicTypeBuilder Struct Reference

A DDS_DynamicTypeBuilder object represents the state of a particular type defined according to the Type System. It is used to instantiate concrete DDS_DynamicType objects.

Related Functions

(Note that these are not member functions.)

- DDS_ReturnCode_t DDS_DynamicTypeBuilder_get_descriptor (DDS_DynamicTypeBuilder dtb, DDS_TypeDescriptor *descriptor)

  This operation provides a summary of the state of this type.
- const char * DDS_DynamicTypeBuilder_get_name (DDS_DynamicTypeBuilder dtb)

  This convenience operation provides the fully qualified name of this type. I.
- DDS_TypeKind DDS_DynamicTypeBuilder_get_kind (DDS_DynamicTypeBuilder dtb)

  This convenience operation indicates the kind of this type (e.g., integer, structure, etc.).
- DDS_ReturnCode_t DDS_DynamicTypeBuilder_get_member_by_name (DDS_DynamicTypeBuilder dtb, DDS_DynamicTypeMember *member, const DDS_ObjectName name)

  This operation accesses a member by name.
- DDS_ReturnCode_t DDS_DynamicTypeBuilder_get_all_members_by_name (DDS_DynamicTypeBuilder dtb, DDS_DynamicTypeMembersByName *member)

  This operation provides access to the 'members_by_name' map.
- DDS_ReturnCode_t DDS_DynamicTypeBuilder_get_member (DDS_DynamicTypeBuilder dtb, DDS_DynamicTypeMember *member, const DDS_MemberId id)

  This operation accesses a member by id.
- DDS_ReturnCode_t DDS_DynamicTypeBuilder_get_all_members (DDS_DynamicTypeBuilder dtb, DDS_DynamicTypeMembersBy *member)

  This operation provides access to the 'members_by_id' map.
- unsigned int DDS_DynamicTypeBuilder_get_annotation_count (DDS_DynamicTypeBuilder dtb)

  Return the number of annotations applied to this type.
- DDS_ReturnCode_t DDS_DynamicTypeBuilder_get_annotation (DDS_DynamicTypeBuilder dtb, DDS_AnnotationDescriptor *descriptor, const unsigned int idx)

  Access an annotation at the specified index. On success, the 'descriptor' parameter is set to the annotation.
- unsigned char DDS_DynamicTypeBuilder_equals (DDS_DynamicTypeBuilder dtb, const DDS_DynamicType other)

  Two types shall be considered equal if and only if all of their respective properties are equal.
- DDS_ReturnCode_t DDS_DynamicTypeBuilder_add_member (DDS_DynamicTypeBuilder dtb, const DDS_MemberDescriptor *descriptor)

  Add a member to the type.
- DDS_ReturnCode_t DDS_DynamicTypeBuilder_apply_annotation (DDS_DynamicTypeBuilder dtb, const DDS_AnnotationDescriptor *descriptor)

  Apply the given annotation to this type.
- DDS_ReturnCode_t DDS_DynamicTypeBuilder_set_extensibility (DDS_DynamicTypeBuilder dtb, uint32_t ext)

  Assign the specified extensibility to the Builder (struct/union)
- DDS_DynamicType DDS_DynamicTypeBuilder_build (DDS_DynamicTypeBuilder dtb)

  Create an immutable DDS_DynamicType object defined by the builder's current state.
- DDS_ReturnCode_t DDS_DynamicTypeBuilder_delete_type (DDS_DynamicTypeBuilder dtb, DDS_DynamicType type)

  Destroy the type information contained in this DDS_DynamicTypeBuilder instance.

10.7.1 Detailed Description

A DDS_DynamicTypeBuilder object represents the state of a particular type defined according to the Type System. It is used to instantiate concrete DDS_DynamicType objects.

See also

DDS_DynamicType
DDS_DynamicTypeBuilderFactory
DDS_DynamicTypeSupport

10.8 DDS_DynamicTypeMember Struct Reference

A DDS_DynamicTypeMember represents a "member" of a type. A "member" in this sense may be a member of an aggregated type, a constant within an enumeration, or some other type substructure.

Related Functions

(Note that these are not member functions.)

- DDS_ReturnCode_t DDS_DynamicTypeMember_get_descriptor (DDS_DynamicTypeMember dtm, DDS_MemberDescriptor *descriptor)

  This operation accesses the current state of this type.
- unsigned int DDS_DynamicTypeMember_get_annotation_count (DDS_DynamicTypeMember dtm)

  Return the number of annotations applied to this type.
- DDS_ReturnCode_t DDS_DynamicTypeMember_get_annotation (DDS_DynamicTypeMember dtm, DDS_AnnotationDescriptor *descriptor, const unsigned int idx)

  Access an annotation at the specified index. On success, the 'descriptor' parameter is set to the annotation.
- unsigned char DDS_DynamicTypeMember_equals (DDS_DynamicTypeMember dtm, const DDS_DynamicTypeMember other)

  Compare two DDS_DynamicTypeMember's.
- DDS_MemberId DDS_DynamicTypeMember_get_id (DDS_DynamicTypeMember dtm)

  This convenience operation provides the member ID of this member.
- const char * DDS_DynamicTypeMember_get_name (DDS_DynamicTypeMember dtm)

  This convenience operation provides the name of this member.
- DDS_MemberFlag DDS_DynamicTypeMember_get_flags (DDS_DynamicTypeMember dtm)

  Access the type flags.
- DDS_ReturnCode_t DDS_DynamicTypeMember_set_flags (DDS_DynamicTypeMember dtm, DDS_MemberFlag f)

  Set the type flags.

10.8.1 Detailed Description

A DDS_DynamicTypeMember represents a "member" of a type. A "member" in this sense may be a member of an aggregated type, a constant within an enumeration, or some other type substructure.

See also

DDS_DynamicType
DDS_DynamicTypeBuilderFactory
DDS_DynamicTypeBuilder

10.9 DDS_DynamicTypeSupport Struct Reference

The DDS_DynamicTypeSupport interface extends the DDS_TypeSupport interface defined by the DDS specification. This TypeSupport operates on DDS_DynamicData samples.

Related Functions

(Note that these are not member functions.)

- DDS_DynamicTypeSupport DDS_DynamicTypeSupport_create_type_support (const DDS_DynamicType type)

  Create and return a new DDS_DynamicTypeSupport object capable of supporting the given type.
- DDS_ReturnCode_t DDS_DynamicTypeSupport_delete_type_support (DDS_DynamicTypeSupport type_support)

  Delete the given type support object, which was previously created by a call to DDS_DynamicTypeSupport_create_type_support.
- DDS_ReturnCode_t DDS_DynamicTypeSupport_register_type (DDS_DynamicTypeSupport dts, const DDS_DomainParticipant participant, const char *type_name)

  Registers the TypeSupport with the DDS_DomainParticipant.
- const char * DDS_DynamicTypeSupport_get_type_name (DDS_DynamicTypeSupport dts)

  Returns the name of the type supported by this TypeSupport instance.
- DDS_DynamicType DDS_DynamicTypeSupport_get_type (DDS_DynamicTypeSupport dts)

  Returns the DDS_DynamicType that is supported by this TypeSupport instance.

10.9.1 Detailed Description

The DDS_DynamicTypeSupport interface extends the DDS_TypeSupport interface defined by the DDS specification. This TypeSupport operates on DDS_DynamicData samples.

See also

DDS_DynamicType
DDS_DynamicTypeBuilderFactory
DDS_DynamicTypeBuilder

10.10 DDS_DynamicType Struct Reference

An instance of DDS_DynamicType represent a type's schema: its physical name, kind, member definitions (if any), and so on.

Related Functions

(Note that these are not member functions.)

- DDS_ReturnCode_t DDS_DynamicType_get_descriptor (DDS_DynamicType dt, DDS_TypeDescriptor *descriptor)

  This operation provides a summary of the current state of this type.
- const char * DDS_DynamicType_get_name (DDS_DynamicType dt)

  This convenience operation provides the fully qualified name of this type.
- DDS_TypeKind DDS_DynamicType_get_kind (DDS_DynamicType dt)

  This convenience operation returns the kind of this type (e.g., integer, structure, etc.).
- DDS_ReturnCode_t DDS_DynamicType_get_member_by_name (DDS_DynamicType dt, DDS_DynamicTypeMember *member, const DDS_ObjectName name)

  This operation accesses a member by name.
- DDS_ReturnCode_t DDS_DynamicType_get_all_members_by_name (DDS_DynamicType dt, DDS_DynamicTypeMembersByNam *member)

  This operation provides access to the 'members_by_name' map.
- DDS_ReturnCode_t DDS_DynamicType_get_member (DDS_DynamicType dt, DDS_DynamicTypeMember *member, const DDS_MemberId id)

  This operation accesses a member by id.
- DDS_ReturnCode_t DDS_DynamicType_get_all_members (DDS_DynamicType dt, DDS_DynamicTypeMembersById *member)

  This operation provides access to the 'members_by_id' map.
- unsigned int DDS_DynamicType_get_annotation_count (DDS_DynamicType dt)

  Return the number of annotations applied to this type.
- DDS_ReturnCode_t DDS_DynamicType_get_annotation (DDS_DynamicType dt, DDS_AnnotationDescriptor *descriptor, const unsigned int idx)

  Access an annotation at the specified index. On success, the 'descriptor' parameter is set to the annotation.
- unsigned char DDS_DynamicType_equals (DDS_DynamicType dt, const DDS_DynamicType other)

  Two types shall be considered equal if and only if all of their respective properties are equal.

10.10.1 Detailed Description

An instance of DDS_DynamicType represent a type's schema: its physical name, kind, member definitions (if any), and so on.

See also

DDS_DynamicTypeSupport
DDS_DynamicTypeBuilderFactory
DDS_DynamicTypeBuilder

10.11 DDS_TypeDescriptor Struct Reference

A DDS_TypeDescriptor comprises the state of a type.

Data Fields

- DDS_TypeKind kind

  the 'kind' of this type
- DDS_ObjectName name

  the 'name' of this type
- DDS_DynamicType base_type

  the 'base_type' of this type
- DDS_DynamicType discriminator_type

  If this descriptor represents a union type, this field indicates the type of the discriminator of the union.
- DDS_BoundSeq bound

  The bound property indicates the bound of collection and similar types.
- DDS_DynamicType element_type

  If this descriptor represents an array, sequence, or string type, this property indicates the element type of the collection.
- DDS_DynamicType key_element_type

  If this descriptor represents a map type, this property indicates the value element type of the map.

Related Functions

(Note that these are not member functions.)

- DDS_ReturnCode_t DDS_TypeDescriptor_copy_from (DDS_TypeDescriptor *to, const DDS_TypeDescriptor *from)

  Overwrite the contents of this descriptor with those of another descriptor.
- unsigned char DDS_TypeDescriptor_equals (const DDS_TypeDescriptor *td, const DDS_TypeDescriptor *other)

  Compare two DDS_TypeDescriptor's.
- unsigned char DDS_TypeDescriptor_is_consistent (const DDS_TypeDescriptor *td)

  Indicates whether the states of all of this descriptor's properties are consistent.

10.11.1 Detailed Description

A DDS_TypeDescriptor comprises the state of a type.

10.11.2 Field Documentation

10.11.2.1 DDS_DynamicType DDS_TypeDescriptor::base_type

the 'base_type' of this type

- If this descriptor represents a structure type, base_type indicates the supertype of that type. A nil value of this property indicates that the structure type has no supertype.

- If this descriptor represents an alias type, base_type indicates the type being aliased. A nil value for this property is not considered consistent.

In all other cases, a consistent descriptor shall have a nil value for this property.

10.11.2.2 DDS_BoundSeq DDS_TypeDescriptor::bound

The bound property indicates the bound of collection and similar types.

- If this descriptor represents an array type, the length of the property value indicates the number of dimensions in the array, and each value indicates the bound of the corresponding dimension.
- If this descriptor represents a sequence, map, bit set, or string type, the length of the property value is one and the integral value in that property indicates the bound of the collection.

In all other cases, a consistent descriptor shall have a nil value for this property.

10.11.2.3 DDS_DynamicType DDS_TypeDescriptor::element_type

If this descriptor represents an array, sequence, or string type, this property indicates the element type of the collection.

It must not be nil for the descriptor to be consistent. If this descriptor represents a map type, this property indicates the value element type of the map. It must not be nil for the descriptor to be consistent.

If this descriptor represents a bit set type, this property must indicate a Boolean type for the descriptor to be consistent.

If this descriptor represents any other kind of type, this property must be nil for the descriptor to be consistent.

10.11.2.4 DDS_DynamicType DDS_TypeDescriptor::key_element_type

If this descriptor represents a map type, this property indicates the value element type of the map.

It must not be nil for the descriptor to be consistent.

If this descriptor represents any other kind of type, this property must be nil for the descriptor to be consistent.

Chapter 11

CoreDX DDS Transports

11.1 CoreDX_LmtTransportConfig Struct Reference

Structure that holds LMT Transport configuration items.

Data Fields

- int so_sndbuf
- int so_rcvbuf
- unsigned int max_tx_size
- unsigned int max_rx_buf_size
- unsigned int debug_flags

11.1.1 Detailed Description

Structure that holds LMT Transport configuration items.

See also

DomainParticipant_add_transport()

11.1.2 Field Documentation

11.1.2.1 unsigned int CoreDX_LmtTransportConfig::debug_flags

adjust the debug output of the transport

11.1.2.2 unsigned int CoreDX_LmtTransportConfig::max_rx_buf_size

limit size of RX buffer (per connection)

11.1.2.3 unsigned int CoreDX_LmtTransportConfig::max_tx_size

largest LMT packet size we transmit (default: 64K, max: 64K)

11.1.2.4 int CoreDX_LmtTransportConfig::so_rcvbuf

size in bytes for socket RCVBUF

11.1.2.5 int CoreDX_LmtTransportConfig::so_sndbuf

size in bytes for socket SNDBUF

11.2 CoreDX_LmtTransport Struct Reference

An instance of a Transport that can be added to a DomainParticipant.

Related Functions

(Note that these are not member functions.)

- DDS_ReturnCode_t CoreDX_LmtTransport_get_default_config (CoreDX_LmtTransportConfig *config)
- DDS_ReturnCode_t CoreDX_LmtTransport_get_env_config (CoreDX_LmtTransportConfig *config)
- DDS_ReturnCode_t CoreDX_LmtTransport_clear_config (CoreDX_LmtTransportConfig *config)
- CoreDX_Transport * CoreDX_LmtTransport_create_transport (CoreDX_LmtTransportConfig *config)

11.2.1 Detailed Description

An instance of a Transport that can be added to a DomainParticipant.

The LmtTransport support RTPS communication through a local machine mechanism for optimized on-platform communications.

See also

DDS_DomainParticipant_add_transport()

11.2.2 Friends And Related Function Documentation

11.2.2.1 DDS_ReturnCode_t CoreDX_LmtTransport_clear_config (CoreDX_LmtTransportConfig * *config*) [related]

Clears any dynamic memory allocated during initialization of the CoreDX_LmtTransportConfig object.

11.2.2.2 CoreDX_Transport * CoreDX_LmtTransport_create_transport (CoreDX_LmtTransportConfig * *config*) [related]

Allocate and initialize a CoreDX_LmtTransport object instance. The returned CoreDX_Transport object can be added to a single DomainParticipant.

Note

The DomainParticipant takes ownership of the provided transport; when the DomainParticipant is deleted, it will release the resources of all added transports.

11.2.2.3 DDS_ReturnCode_t CoreDX_LmtTransport_get_default_config (CoreDX_LmtTransportConfig * *config*) [related]

Initialize the LmtTransportConfig object with default values. Currently assigned values may be overwritten by defaults.

11.2.2.4 DDS_ReturnCode_t CoreDX_LmtTransport_get_env_config (CoreDX_LmtTransportConfig * *config*) [related]

Query for environment variables that impact lmt transport configuration. Load the values (if any) into the LmtTransport-Config object. Currently assigned values may be overwritten by values derived from environment variables.

11.3 CoreDX_TcpTransportConfig Struct Reference

Structure that holds TCP Transport configuration items.

Data Fields

- short participant_index
- CoreDX_IpTransportInterfaceSeq interfaces
- unsigned char dynamic_interfaces
- int tx_max_packet_size
- unsigned int debug_flags

11.3.1 Detailed Description

Structure that holds TCP Transport configuration items.

See also

DomainParticipant::add_transport(Transport) add_transport()

11.3.2 Field Documentation

11.3.2.1 unsigned int CoreDX_TcpTransportConfig::debug_flags

adjust the debug output of the transport

11.3.2.2 unsigned char CoreDX_TcpTransportConfig::dynamic_interfaces

detect and handle changes to interface addresses

11.3.2.3 CoreDX_IpTransportInterfaceSeq CoreDX_TcpTransportConfig::interfaces

default: empty -> use all available interfaces

11.3.2.4 short CoreDX_TcpTransportConfig::participant_index

-1: auto detect; else force (may fail if another participant is using the ports (can't exceed 120)

11.3.2.5 int CoreDX_TcpTransportConfig::tx_max_packet_size

default: 64K (to match the udp limit)

11.4 CoreDX_TcpTransport Struct Reference

An instance of a Transport that can be added to a DomainParticipant.

Related Functions

(Note that these are not member functions.)

- DDS_ReturnCode_t CoreDX_TcpTransport_get_default_config (CoreDX_TcpTransportConfig *config)
- DDS_ReturnCode_t CoreDX_TcpTransport_get_env_config (CoreDX_TcpTransportConfig *config)
- DDS_ReturnCode_t CoreDX_TcpTransport_clear_config (CoreDX_TcpTransportConfig *config)
- CoreDX_Transport * CoreDX_TcpTransport_create_transport (CoreDX_TcpTransportConfig *config)

11.4.1 Detailed Description

An instance of a Transport that can be added to a DomainParticipant.

The TcpTransport support RTPS over a TCP/IP network. It can be used for both on-platform and off-platform communications.

See also

DDS_DomainParticipant_add_transport()

11.4.2 Friends And Related Function Documentation

11.4.2.1 DDS_ReturnCode_t CoreDX_TcpTransport_clear_config (CoreDX_TcpTransportConfig * *config*) [related]

Clears any dynamic memory allocated during initialization of the CoreDX_TcpTransportConfig object.

11.4.2.2 CoreDX_Transport * CoreDX_TcpTransport_create_transport (CoreDX_TcpTransportConfig * *config*) [related]

Allocate and initialize a CoreDX_TcpTransport object instance. The returned CoreDX_Transport object can be added to a single DomainParticipant.

Note

The DomainParticipant takes ownership of the provided transport; when the DomainParticipant is deleted, it will release the resources of all added transports.

11.4.2.3 DDS_ReturnCode_t CoreDX_TcpTransport_get_default_config (CoreDX_TcpTransportConfig * *config*) [related]

Initialize the TcpTransportConfig object with default values. Currently assigned values may be overwritten by defaults.

11.4.2.4 DDS_ReturnCode_t CoreDX_TcpTransport_get_env_config (CoreDX_TcpTransportConfig * *config*) [related]

Query for environment variables that impact tcp transport configuration. Load the values (if any) into the TcpTransportConfig object. Currently assigned values may be overwritten by values derived from environment variables.

11.5 CoreDX_UdpTransportConfig Struct Reference

Structure that holds UDP Transport configuration items.

Data Fields

- short participant_index
- unsigned char use_ipv4
- unsigned char use_ipv6
- CoreDX_IpTransportInterfaceSeq interfaces
- unsigned char dynamic_interfaces
- int rx_init_buffer_size
- int rx_max_buffer_size
- int tx_max_packet_size
- int so_rcvbuf
- int so_sndbuf
- unsigned char meta_multicast_address_v4 [4]
- unsigned char user_multicast_address_v4 [4]
- unsigned char meta_multicast_address_v6 [16]
- unsigned char user_multicast_address_v6 [16]
- unsigned char multicast_ttl
- unsigned char tx_meta_multicast
- unsigned char tx_meta_unicast
- unsigned char rx_meta_multicast
- unsigned char rx_user_multicast
- unsigned char advertise_meta_multicast
- unsigned char advertise_user_multicast
- unsigned char broadcast_address [4]
- unsigned char do_meta_broadcast
- unsigned int debug_flags

11.5.1 Detailed Description

Structure that holds UDP Transport configuration items.

See also

DomainParticipant_add_transport()

11.5.2 Field Documentation

11.5.2.1 unsigned char CoreDX_UdpTransportConfig::advertise_meta_multicast

advertise we can RX META MULTICAST

11.5.2.2 unsigned char CoreDX_UdpTransportConfig::advertise_user_multicast

advertise we can RX USER MULTICAST

11.5.2.3 unsigned char CoreDX_UdpTransportConfig::broadcast_address[4]

default: 255.255.255.255

11.5.2.4 unsigned int CoreDX_UdpTransportConfig::debug_flags

adjust the debug output from the transport

11.5.2.5 unsigned char CoreDX_UdpTransportConfig::do_meta_broadcast

enable broadcast of META (DPD discovery) data default: 0 (off)

11.5.2.6 unsigned char CoreDX_UdpTransportConfig::dynamic_interfaces

detect and handle changes to interface addresses

11.5.2.7 CoreDX_IpTransportInterfaceSeq CoreDX_UdpTransportConfig::interfaces

default: empty -> use all available interfaces

11.5.2.8 unsigned char CoreDX_UdpTransportConfig::meta_multicast_address_v4[4]

default: [239 255 0 1] per the standard

11.5.2.9 unsigned char CoreDX_UdpTransportConfig::meta_multicast_address_v6[16]

default: [ff03:0000:0000:0000:0000:efff:0001]

11.5.2.10 unsigned char CoreDX_UdpTransportConfig::multicast_ttl

default: 1 (0: disable all MCAST TX)

11.5.2.11 short CoreDX_UdpTransportConfig::participant_index

-1: auto detect; else force (may fail if another participant is using the ports (can't exceed 120)

11.5.2.12 int CoreDX_UdpTransportConfig::rx_init_buffer_size

initial size of data buffer

11.5.2.13 int CoreDX_UdpTransportConfig::rx_max_buffer_size

maximum size of data buffer

11.5.2.14 unsigned char CoreDX_UdpTransportConfig::rx_meta_multicast

enable META MULTICAST (discovery) RX

11.5.2.15 unsigned char CoreDX_UdpTransportConfig::rx_user_multicast

enable USER MULTICAST (data) RX

11.5.2.16 int CoreDX_UdpTransportConfig::so_rcvbuf

socket RCVBUF size (set to -1 to use OS default)

11.5.2.17 int CoreDX_UdpTransportConfig::so_sndbuf

socket SNDBUF size (set to -1 to use OS default)

11.5.2.18 int CoreDX_UdpTransportConfig::tx_max_packet_size

default: 64K (udp limit)

11.5.2.19 unsigned char CoreDX_UdpTransportConfig::tx_meta_multicast

enable META MULTICAST (discovery) TX

11.5.2.20 unsigned char CoreDX_UdpTransportConfig::tx_meta_unicast

enable META UNICAST (discovery) TX

11.5.2.21 unsigned char CoreDX_UdpTransportConfig::use_ipv4

Support IPv4 communications (default ON (1))

11.5.2.22 unsigned char CoreDX_UdpTransportConfig::use_ipv6

Support IPv4 communications (default OFF(0))

11.5.2.23 unsigned char CoreDX_UdpTransportConfig::user_multicast_address_v4[4]

default: [239 255 0 1] per the standard

11.5.2.24 unsigned char CoreDX_UdpTransportConfig::user_multicast_address_v6[16]

default: [ff03:0000:0000:0000:0000:efff:0001]

11.6 CoreDX_UdpTransport Struct Reference

An instance of a Transport that can be added to a DomainParticipant.

Related Functions

(Note that these are not member functions.)

- DDS_ReturnCode_t CoreDX_UdpTransport_get_default_config (CoreDX_UdpTransportConfig *config)
- DDS_ReturnCode_t CoreDX_UdpTransport_get_env_config (CoreDX_UdpTransportConfig *config)
- DDS_ReturnCode_t CoreDX_UdpTransport_clear_config (CoreDX_UdpTransportConfig *config)
- CoreDX_Transport * CoreDX_UdpTransport_create_transport (CoreDX_UdpTransportConfig *config)

11.6.1 Detailed Description

An instance of a Transport that can be added to a DomainParticipant.

The UdpTransport is the default transport used by CoreDX DDS for communications. It support RTPS over a UDP/IP network. It can be used for both on-platform and off-platform communications.

See also

DDS_DomainParticipant_add_transport()

11.6.2 Friends And Related Function Documentation

11.6.2.1 DDS_ReturnCode_t CoreDX_UdpTransport_clear_config (CoreDX_UdpTransportConfig * *config*) [related]

Clears any dynamic memory allocated during initialization of the CoreDX_UdpTransportConfig object.

11.6.2.2 CoreDX_Transport * CoreDX_UdpTransport_create_transport (CoreDX_UdpTransportConfig * *config*) [related]

Allocate and initialize a CoreDX_UdpTransport object instance. The returned CoreDX_Transport object can be added to a single DomainParticipant.

Note

The DomainParticipant takes ownership of the provided transport; when the DomainParticipant is deleted, it will release the resources of all added transports.

11.6.2.3 DDS_ReturnCode_t CoreDX_UdpTransport_get_default_config (CoreDX_UdpTransportConfig * *config*) [related]

Initialize the UdpTransportConfig object with default values. Currently assigned values may be overwritten by defaults.

11.6.2.4 DDS_ReturnCode_t CoreDX_UdpTransport_get_env_config (CoreDX_UdpTransportConfig * *config*) [related]

Query for environment variables that impact udp transport configuration. Load the values (if any) into the UdpTransportConfig object. Currently assigned values may be overwritten by values derived from environment variables.

Chapter 12

CoreDX DDS DynamicTypes (Deprecated)

NOTE: This API is deprecated. The preferred DynamicType API is the one provided by the XTypes implementation.

12.1 CDX_DynamicType Struct Reference

CDX_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 CDX_DynamicType_get_type (CDX_DynamicType t)

  Provides access to the 'type' of the DynamicType object. Applicable to any DynamicType.
- unsigned char CDX_DynamicType_get_octet (CDX_DynamicType t)

  Provides access to data held in an OCTET DynamicType object.
- unsigned char CDX_DynamicType_get_boolean (CDX_DynamicType t)

  Provides access to data held in an BOOLEAN DynamicType object.
- char CDX_DynamicType_get_char (CDX_DynamicType t)

  Provides access to data held in an CHAR DynamicType object.
- int16_t CDX_DynamicType_get_short (CDX_DynamicType t)

  Provides access to data held in an SHORT DynamicType object.
- uint16_t CDX_DynamicType_get_ushort (CDX_DynamicType t)

  Provides access to data held in an UNSIGNED SHORT DynamicType object.
- int32_t CDX_DynamicType_get_long (CDX_DynamicType t)

  Provides access to data held in an LONG DynamicType object.
- uint32_t CDX_DynamicType_get_ulong (CDX_DynamicType t)

  Provides access to data held in an UNSIGNED LONG DynamicType object.
- int64_t CDX_DynamicType_get_longlong (CDX_DynamicType t)

  Provides access to data held in an LONG LONG DynamicType object.
- uint64_t CDX_DynamicType_get_ulonglong (CDX_DynamicType t)

  Provides access to data held in an UNSIGNED LONG LONG DynamicType object.
- float CDX_DynamicType_get_float (CDX_DynamicType t)

Provides access to data held in an FLOAT DynamicType object.

- double CDX_DynamicType_get_double (CDX_DynamicType t)

  Provides access to data held in an DOUBLE DynamicType object.

- int32_t CDX_DynamicType_enum_get_size (CDX_DynamicType t)

  Gets the 'size' of this ENUM type in bytes (either 4 bytes or 2 bytes).

- int32_t CDX_DynamicType_enum_get_num_constants (CDX_DynamicType t)

  Provides access to the number of constants defined for this ENUM type.

- void CDX_DynamicType_EnumConstant_delete (CDX_DynamicType_EnumConstant_t *ec)

  Provides method to reclaim memory held by a CDX_DynamicType_EnumConstant_t. Use this to reclaim the memory returned by one of the following routines: CDX_DynamicType_enum_get_constant() CDX_DynamicType_enum_get_constant_by_name() CDX_DynamicType_enum_get_constant_by_value()

- CDX_DynamicType_EnumConstant_t * CDX_DynamicType_enum_get_constant (CDX_DynamicType t, int32_t index)

  Provides access to a specific constant (name,value pair) within the ENUM type.

- CDX_DynamicType_EnumConstant_t * CDX_DynamicType_enum_get_constant_by_name (CDX_DynamicType t, const char *name)

  Provides access EnumConstant associated with an 'name' in the provided ENUM.

- CDX_DynamicType_EnumConstant_t * CDX_DynamicType_enum_get_constant_by_value (CDX_DynamicType t, uint32_t val)

  Provides access to the EnumConstant associated with a 'value' in the provided ENUM.

- uint32_t CDX_DynamicType_enum_get_value (CDX_DynamicType t)

  Provides access to the 'value' of an instance of type ENUM.

- int32_t CDX_DynamicType_bitset_get_size (CDX_DynamicType t)

  Gets the 'size' of this BITSET type in bytes (either 8, 4, 2, or 1 bytes)

- uint64_t CDX_DynamicType_bitset_get_value (CDX_DynamicType t)

  Provides access to the 'value' of an instance of type BITSET.

- int32_t CDX_DynamicType_bitset_get_num_flags (CDX_DynamicType t)

  Provides access to the number of flags defined for this BITSET type.

- void CDX_DynamicType_BitsetFlag_delete (CDX_DynamicType_BitsetFlag_t *bf)

  Provides method to reclaim memory held by a CDX_DynamicType_BitsetFlag_t. Use this to reclaim the memory returned by one of the following routines: CDX_DynamicType_bitset_get_flag() CDX_DynamicType_bitset_get_flag_by_name()

- CDX_DynamicType_BitsetFlag_t * CDX_DynamicType_bitset_get_flag (CDX_DynamicType t, int32_t index)

  Provides access to a specific flag (name,value pair) within the BITSET type.

- CDX_DynamicType_BitsetFlag_t * CDX_DynamicType_bitset_get_flag_by_name (CDX_DynamicType t, const char *name)

  Provides access BitsetFlag associated with an 'name' in the provided BITSET.

- const char * CDX_DynamicType_get_string (CDX_DynamicType t)

  Provides access to data held in an STRING DynamicType object.

- uint32_t CDX_DynamicType_get_max_length (CDX_DynamicType t)

  Provides access to the maximum length of a DynamicType object.

- uint32_t CDX_DynamicType_get_length (CDX_DynamicType t)

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

- CDX_DynamicType CDX_DynamicType_get_element_type (CDX_DynamicType t)

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

- CDX_DynamicType CDX_DynamicType_get_element (CDX_DynamicType t, uint32_t n)

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

- uint32_t CDX_DynamicType_get_num_fields (CDX_DynamicType t)

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

- CDX_DynamicType CDX_DynamicType_get_field (CDX_DynamicType t, uint32_t n)

  Provides access to a field held by a STRUCT or UNION DynamicType object.
- const char * CDX_DynamicType_get_field_name (CDX_DynamicType t, uint32_t n)

  Provides access to the name of a field held by a STRUCT or UNION DynamicType object.
- unsigned char CDX_DynamicType_get_field_key (CDX_DynamicType t, uint32_t n)

  Provides access to the 'key' indication for a field held by a STRUCT DynamicType object.
- CDX_DynamicType CDX_DynamicType_get_discriminator (CDX_DynamicType t)

  Provides access to the 'discriminator' type of a UNION DynamicType object.
- int32_t CDX_DynamicType_get_default_field (CDX_DynamicType t)

  Provides access to the 'default' field of a UNION DynamicType object.
- uint32_t CDX_DynamicType_get_field_num_labels (CDX_DynamicType t, uint32_t field)

  Provides access to the number of labels assigned to a field in the UNION DynamicType object.
- int32_t CDX_DynamicType_get_field_label (CDX_DynamicType t, uint32_t field, uint32_t label_idx)

  Provides access to a label value assigned to a field in the UNION DynamicType object.
- CDX_DynamicType CDX_DynamicType_get_selected_field (CDX_DynamicType t)

  Provides access to the selected field of a UNION DynamicType object.
- CDX_DynamicType CDX_DynamicType_alloc (DDS_TypeCodeKind type_code)

  Allocates and returns a DynamicType configured to hold the specified 'type_code' type.
- CDX_DynamicType CDX_DynamicType_alloc_basic (DDS_TypeCodeKind type_code)

  Allocates and returns a DynamicType configured to hold the specified 'type_code' basic type.
- CDX_DynamicType CDX_DynamicType_alloc_enum ()

  Allocates and returns an ENUM DynamicType.
- CDX_DynamicType CDX_DynamicType_alloc_bitset ()

  Allocates and returns a BITSET DynamicType.
- CDX_DynamicType CDX_DynamicType_alloc_string ()

  Allocates and returns a STRING DynamicType.
- CDX_DynamicType CDX_DynamicType_alloc_array ()

  Allocates and returns an ARRAY DynamicType.
- CDX_DynamicType CDX_DynamicType_alloc_sequence ()

  Allocates and returns a SEQUENCE DynamicType.
- CDX_DynamicType CDX_DynamicType_alloc_struct ()

  Allocates and returns a STRUCT DynamicType.
- CDX_DynamicType CDX_DynamicType_alloc_union ()

  Allocates and returns a UNION DynamicType.
- void CDX_DynamicType_free (CDX_DynamicType t)

  Reclaims all memory used by an allocated DynamicType object.
- DDS_ReturnCode_t CDX_DynamicType_set_octet (CDX_DynamicType t, unsigned char c)

  Assigns a value to the provided OCTET DynamicType.
- DDS_ReturnCode_t CDX_DynamicType_set_boolean (CDX_DynamicType t, unsigned char c)

  Assigns a value to the provided BOOLEAN DynamicType.
- DDS_ReturnCode_t CDX_DynamicType_set_char (CDX_DynamicType t, char c)

  Assigns a value to the provided CHAR DynamicType.
- DDS_ReturnCode_t CDX_DynamicType_set_short (CDX_DynamicType t, short c)

  Assigns a value to the provided SHORT DynamicType.
- DDS_ReturnCode_t CDX_DynamicType_set_ushort (CDX_DynamicType t, unsigned short c)

  Assigns a value to the provided USHORT DynamicType.
- DDS_ReturnCode_t CDX_DynamicType_set_long (CDX_DynamicType t, long c)

Assigns a value to the provided LONG DynamicType.

- DDS_ReturnCode_t CDX_DynamicType_set_ulong (CDX_DynamicType t, unsigned long c)

  Assigns a value to the provided ULONG DynamicType.
- DDS_ReturnCode_t CDX_DynamicType_set_longlong (CDX_DynamicType t, int64_t c)

  Assigns a value to the provided LONGLONG DynamicType.
- DDS_ReturnCode_t CDX_DynamicType_set_ulonglong (CDX_DynamicType t, uint64_t c)

  Assigns a value to the provided ULONGLONG DynamicType.
- DDS_ReturnCode_t CDX_DynamicType_set_float (CDX_DynamicType t, float c)

  Assigns a value to the provided FLOAT DynamicType.
- DDS_ReturnCode_t CDX_DynamicType_set_double (CDX_DynamicType t, double c)

  Assigns a value to the provided DOUBLE DynamicType.
- DDS_ReturnCode_t CDX_DynamicType_enum_set_size (CDX_DynamicType t, int32_t size)

  Sets the 'size' of this ENUM type in bytes (either 4 bytes or 2 bytes).
- DDS_ReturnCode_t CDX_DynamicType_enum_set_num_constants (CDX_DynamicType t, int32_t num)

  Sets the number of constants defined by this ENUM type.
- DDS_ReturnCode_t CDX_DynamicType_enum_set_constant (CDX_DynamicType t, int32_t n, const char ∗name, uint32_t val)

  Assigns a 'name' to a 'value' in the provided ENUM DynamicType.
- DDS_ReturnCode_t CDX_DynamicType_enum_set_value (CDX_DynamicType t, uint32_t val)

  Assigns a value to this instance of an ENUM.
- DDS_ReturnCode_t CDX_DynamicType_bitset_set_size (CDX_DynamicType t, int32_t size)

  Sets the 'size' of this BITSET type in bytes (either 8, 4, 2, or 1 bytes).
- DDS_ReturnCode_t CDX_DynamicType_bitset_set_num_flags (CDX_DynamicType t, int32_t num)

  Sets the number of flags defined by this BITSET type.
- DDS_ReturnCode_t CDX_DynamicType_bitset_set_flag (CDX_DynamicType t, int32_t n, const char ∗name, uint64_t val)

  Assigns a 'name' to a 'value' in the provided BITSET DynamicType.
- DDS_ReturnCode_t CDX_DynamicType_bitset_set_value (CDX_DynamicType t, uint64_t val)

  Assigns a value to this instance of an BITSET.
- DDS_ReturnCode_t CDX_DynamicType_set_string (CDX_DynamicType t, const char ∗c)

  Assigns a value to the provided STRING DynamicType.
- DDS_ReturnCode_t CDX_DynamicType_set_max_length (CDX_DynamicType t, uint32_t n)

  Assigns a 'max_length' value to the provided STRING, ARRAY, or SEQUENCE DynamicType.
- DDS_ReturnCode_t CDX_DynamicType_set_length (CDX_DynamicType t, uint32_t n)

  Assigns a 'length' value to the provided ARRAY or SEQUENCE DynamicType.
- DDS_ReturnCode_t CDX_DynamicType_set_element_type (CDX_DynamicType t, CDX_DynamicType e)

  Defines the element type of an ARRAY or SEQUENCE.
- DDS_ReturnCode_t CDX_DynamicType_set_element (CDX_DynamicType t, uint32_t n, CDX_DynamicType e)

  Assigns a value to an element of an ARRAY or SEQUENCE.
- DDS_ReturnCode_t CDX_DynamicType_set_num_fields (CDX_DynamicType t, uint32_t n)

  Defines the number of fields held by a STRUCT or UNION DynamicType object.
- DDS_ReturnCode_t CDX_DynamicType_set_field (CDX_DynamicType t, uint32_t n, const char ∗field_name, CDX_DynamicType e, unsigned char key)

  Assigns a value to a field of a STRUCT or UNION DynamicType object.
- DDS_ReturnCode_t CDX_DynamicType_set_discriminator (CDX_DynamicType t, CDX_DynamicType d)

  Defines the type and value of the UNION discriminator.
- DDS_ReturnCode_t CDX_DynamicType_set_default_field (CDX_DynamicType t, int field)

Defines the index of the default field within the UNION.

- DDS_ReturnCode_t CDX_DynamicType_set_field_num_labels (CDX_DynamicType t, uint32_t field, uint32_t n)

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

- DDS_ReturnCode_t CDX_DynamicType_set_field_label (CDX_DynamicType t, uint32_t field, uint32_t label, int32_t val)

  Assigns a label to the specified field within the UNION.

12.1.1 Detailed Description

CDX_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.

12.1.2 Friends And Related Function Documentation

12.1.2.1 CDX_DynamicType CDX_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.

12.1.2.2 CDX_DynamicType CDX_DynamicType_alloc_array ( ) [related]

Allocates and returns an ARRAY DynamicType.

The type of the array elements must be specified by calling CDX_DynamicType_set_element_type(). The size of the array must be specified by calling CDX_DynamicType_set_max_length(). Before adding data to the array, memory for the elements must be allocated by calling CDX_DynamicType_set_length(). Data can be added to the array by calling CDX_DynamicType_set_element().

12.1.2.3 CDX_DynamicType CDX_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, or OCTET.

12.1.2.4 CDX_DynamicType CDX_DynamicType_alloc_bitset ( ) [related]

Allocates and returns a BITSET DynamicType.

By default the bitset has zero flags defined (that is, [name,value] pairs). CDX_DynamicType_bitset_set_flag() is used to add (name, value) pairs to the BITSET. CDX_DynamicType_bitset_get_flag_by_name() is useful for querying the 'value' associated with a specified 'name'. Use CDX_DynamicType_bitset_set_value() to set the value of a specific BITSET instance. This creates an BITSET with a default BitBound of 32 bits (4 bytes).

12.1.2.5 CDX_DynamicType CDX_DynamicType_alloc_enum ( ) [related]

Allocates and returns an ENUM DynamicType.

By default the enum has zero (name,value) pairs. CDX_DynamicType_enum_set_constant() is used to add (name, value) pairs to the ENUM. CDX_DynamicType_enum_get_constant_by_name() is useful for querying the 'value' associated with a specified 'name'. CDX_DynamicType_enum_get_constant_by_value() is useful for querying the 'name' associated with a specified 'value'. Use CDX_DynamicType_enum_set_value() to set the value of a specific ENUM instance. This creates an enum with a default size of 32 bits (4 bytes). Use CDX_DynamicType_set_enum_size() to change the 'size' of this ENUM type.

12.1.2.6 CDX_DynamicType CDX_DynamicType_alloc_sequence ( ) [related]

Allocates and returns a SEQUENCE DynamicType.

The type of the sequence elements must be specified by calling CDX_DynamicType_set_element_type(). The size of the sequence must be specified by by calling CDX_DynamicType_set_max_length(). Before adding data to the sequence, memory for the elements must be allocated by calling CDX_DynamicType_set_length(). Data can be added to the sequence by calling CDX_DynamicType_set_element().

12.1.2.7 CDX_DynamicType CDX_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 CDX_DynamicType_set_max_length().

12.1.2.8 CDX_DynamicType CDX_DynamicType_alloc_struct ( ) [related]

Allocates and returns a STRUCT DynamicType.

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

12.1.2.9 CDX_DynamicType CDX_DynamicType_alloc_union ( ) [related]

Allocates and returns a UNION DynamicType.

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

1. CDX_DynamicType_set_field() must be used to define the type of the field.
2. The CDX_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 CDX_DynamicType_set_field_label() function must be used to define a specific case label for the field.

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

12.1.2.10 CDX_DynamicType_BitsetFlag_t * CDX_DynamicType_bitset_get_flag (CDX_DynamicType *t*, int32_t *index*) [related]

Provides access to a specific flag (name,value pair) within the BITSET type.

Note

Caller is responsible for freeing the returned pointer with CDX_DynamicType_BitsetFlag_delete().

Return values

An	allocated BitsetFlag structure
NULL	if the 'name' is not found.

12.1.2.11 CDX_DynamicType_BitsetFlag_t * CDX_DynamicType_bitset_get_flag_by_name (CDX_DynamicType *t*, const char * *name*) [related]

Provides access BitsetFlag associated with an 'name' in the provided BITSET.

Note

Caller is responsible for freeing the returned pointer with CDX_DynamicType_BitsetFlag_delete().

Return values

An	allocated BitsetFlag structure
NULL	if the 'name' is not found.

12.1.2.12 int32_t CDX_DynamicType_bitset_get_num_flags (CDX_DynamicType *t*) [related]

Provides access to the number of flags defined for this BITSET type.

Return values

-1	if the DynamicType is not of BITSET type.
number	of flags defined in this BITSET.

12.1.2.13 int32_t CDX_DynamicType_bitset_get_size (CDX_DynamicType *t*) [related]

Gets the 'size' of this BITSET type in bytes (either 8, 4, 2, or 1 bytes)

Return values

-1	if the DynamicType is not of type BITSET.
number	of bytes used to represent this BITSET type.

12.1.2.14 DDS_ReturnCode_t CDX_DynamicType_bitset_set_flag (CDX_DynamicType *t,* int32_t *n,* const char * *name,* uint64_t *val*) [related]

Assigns a 'name' to a 'value' in the provided BITSET DynamicType.

Return values

BAD_PARAMETER	if 't' is not of type BITSET.
BAD_PARAMETER	if a flag with 'name' already exists with a different 'val'
OUT_OF_RESOURCES	if unable to add new flag
OK	upon success.

12.1.2.15 DDS_ReturnCode_t CDX_DynamicType_bitset_set_num_flags (CDX_DynamicType *t,* int32_t *num*) [related]

Sets the number of flags defined by this BITSET type.

Return values

BAD_PARAMETER	if 't' is not of type BITSET.
OUT_OF_RESOURCES	if unable to configure requested number of flags.
OK	upon success.

12.1.2.16 DDS_ReturnCode_t CDX_DynamicType_bitset_set_size (CDX_DynamicType *t,* int32_t *size*) [related]

Sets the 'size' of this BITSET type in bytes (either 8, 4, 2, or 1 bytes).

Return values

BAD_PARAMETER	if 't' is not of type BITSET, or size is not one of 8, 4, 2, or 1.
OK	upon success.

12.1.2.17 DDS_ReturnCode_t CDX_DynamicType_bitset_set_value (CDX_DynamicType *t,* uint64_t *val*) [related]

Assigns a value to this instance of an BITSET.

Return values

BAD_PARAMETER	if 't' is not of type BITSET, or if 'val' requires more bits than supported by this BITSET.
OK	upon success.

12.1.2.18 CDX_DynamicType_EnumConstant_t * CDX_DynamicType_enum_get_constant (CDX_DynamicType *t*, int32_t *index*) [related]

Provides access to a specific constant (name,value pair) within the ENUM type.

Note

Caller is responsible for freeing the returned pointer with CDX_DynamicType_EnumConstant_delete().

Return values

An	allocated EnumConstant structure
NULL	if the 'name' is not found.

12.1.2.19 CDX_DynamicType_EnumConstant_t * CDX_DynamicType_enum_get_constant_by_name (CDX_DynamicType *t*, const char * *name*) [related]

Provides access EnumConstant associated with an 'name' in the provided ENUM.

Note

Caller is responsible for freeing the returned pointer with CDX_DynamicType_EnumConstant_delete().

Return values

An	allocated EnumConstant structure
NULL	if the 'name' is not found.

12.1.2.20 CDX_DynamicType_EnumConstant_t * CDX_DynamicType_enum_get_constant_by_value (CDX_DynamicType *t*, uint32_t *val*) [related]

Provides access to the EnumConstant associated with a 'value' in the provided ENUM.

Note

Caller is responsible for freeing the returned pointer with CDX_DynamicType_EnumConstant_delete().

Return values

An	allocated EnumConstant structure
NULL	if the 'value' is not found.

12.1.2.21 int32_t CDX_DynamicType_enum_get_num_constants (CDX_DynamicType *t*) [related]

Provides access to the number of constants defined for this ENUM type.

Return values

-1	if the DynamicType is not of ENUM type.
number	of constants defined in this ENUM.

12.1.2.22 int32_t CDX_DynamicType_enum_get_size (CDX_DynamicType *t*) [related]

Gets the 'size' of this ENUM type in bytes (either 4 bytes or 2 bytes).

Return values

-1	if the DynamicType is not of type ENUM.
number	of bytes used to represent this ENUM type.

12.1.2.23 DDS_ReturnCode_t CDX_DynamicType_enum_set_constant (CDX_DynamicType *t,* int32_t *n,* const char * *name,* uint32_t *val*) [related]

Assigns a 'name' to a 'value' in the provided ENUM DynamicType.

Return values

BAD_PARAMETER	if 't' is not of type ENUM.
BAD_PARAMETER	if a constant with 'name' already exists with a different 'val'
BAD_PARAMETER	if a constant with 'val' already exists with a different 'name'
OUT_OF_RESOURCES	if unable to add new value
OK	upon success.

12.1.2.24 DDS_ReturnCode_t CDX_DynamicType_enum_set_num_constants (CDX_DynamicType *t,* int32_t *num*) [related]

Sets the number of constants defined by this ENUM type.

Return values

BAD_PARAMETER	if 't' is not of type ENUM.
OUT_OF_RESOURCES	if unable to configure requested number of constants
OK	upon success.

12.1.2.25 DDS_ReturnCode_t CDX_DynamicType_enum_set_size (CDX_DynamicType *t,* int32_t *size*) [related]

Sets the 'size' of this ENUM type in bytes (either 4 bytes or 2 bytes).

Return values

BAD_PARAMETER	if 't' is not of type ENUM, or size is not 4 or 2.

Return values

OK	upon success.

12.1.2.26 DDS_ReturnCode_t CDX_DynamicType_enum_set_value (CDX_DynamicType *t,* uint32_t *val*) [related]

Assigns a value to this instance of an ENUM.

Return values

BAD_PARAMETER	if 't' is not of type ENUM.
OK	upon success.

12.1.2.27 void CDX_DynamicType_free (CDX_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 CDX_DynamicType_alloc_xxx() routines or by CDX_TypeDefinition_create_dynamictype(). This routines frees the type information as well as any data values contained in the DynamicType object.

12.1.2.28 unsigned char CDX_DynamicType_get_boolean (CDX_DynamicType *t*) [related]

Provides access to data held in an BOOLEAN DynamicType object.

Return values

unsigned_char	0 indicates FALSE, non-zero indicates TRUE.

12.1.2.29 char CDX_DynamicType_get_char (CDX_DynamicType *t*) [related]

Provides access to data held in an CHAR DynamicType object.

Return values

char	the data value held by 't'.

12.1.2.30 int32_t CDX_DynamicType_get_default_field (CDX_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.

12.1.2.31 CDX_DynamicType CDX_DynamicType_get_discriminator (CDX_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

CDX_DynamicType	the discriminator data.

12.1.2.32 double CDX_DynamicType_get_double (CDX_DynamicType *t*) [related]

Provides access to data held in an DOUBLE DynamicType object.

Return values

double	the data value held by 't'.

12.1.2.33 CDX_DynamicType CDX_DynamicType_get_element (CDX_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'.

12.1.2.34 CDX_DynamicType CDX_DynamicType_get_element_type (CDX_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'.

12.1.2.35 CDX_DynamicType CDX_DynamicType_get_field (CDX_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

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

12.1.2.36 unsigned char CDX_DynamicType_get_field_key (CDX_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.

12.1.2.37 int32_t CDX_DynamicType_get_field_label (CDX_DynamicType *t*, uint32_t *field*, uint32_t *label_idx*) [related]

Provides access to a label value assigned to a field in the UNION DynamicType object.

This is applicable only for a UNION DynamicType object. This the value of the indicated label associated with a particular field in the UNION. [The 'default:' label is not included in these values - as a result, it is possible for one field to have 'zero' labels.]

Return values

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

12.1.2.38 const char * CDX_DynamicType_get_field_name (CDX_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'.

12.1.2.39 uint32_t CDX_DynamicType_get_field_num_labels (CDX_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'.

12.1.2.40 float CDX_DynamicType_get_float (CDX_DynamicType *t*) [related]

Provides access to data held in an FLOAT DynamicType object.

Return values

float	the data value held by 't'.

12.1.2.41 uint32_t CDX_DynamicType_get_length (CDX_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'.

12.1.2.42 int32_t CDX_DynamicType_get_long (CDX_DynamicType *t*) [related]

Provides access to data held in an LONG DynamicType object.

Return values

int32_t	the data value held by 't'.

12.1.2.43 int64_t CDX_DynamicType_get_longlong (CDX_DynamicType *t*) [related]

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

Return values

int64_t	the data value held by 't'.

12.1.2.44 uint32_t CDX_DynamicType_get_max_length (CDX_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'.

12.1.2.45 uint32_t CDX_DynamicType_get_num_fields (CDX_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'.

12.1.2.46 unsigned char CDX_DynamicType_get_octet (CDX_DynamicType *t*) [related]

Provides access to data held in an OCTET DynamicType object.

Return values

unsigned_char	the data value held by 't'.

12.1.2.47 CDX_DynamicType CDX_DynamicType_get_selected_field (CDX_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'.

12.1.2.48 int16_t CDX_DynamicType_get_short (CDX_DynamicType *t*) [related]

Provides access to data held in an SHORT DynamicType object.

Return values

int16_t	the data value held by 't'.

12.1.2.49 const char * CDX_DynamicType_get_string (CDX_DynamicType *t*) [related]

Provides access to data held in an STRING DynamicType object.

Return values

*const_char_**	the data value held by 't'.

12.1.2.50 DDS_TypeCodeKind CDX_DynamicType_get_type (CDX_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'.

12.1.2.51 uint32_t CDX_DynamicType_get_ulong (CDX_DynamicType *t*) [related]

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

Return values

uint32_t	the data value held by 't'.

12.1.2.52 uint64_t CDX_DynamicType_get_ulonglong (CDX_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'.

12.1.2.53 uint16_t CDX_DynamicType_get_ushort (CDX_DynamicType *t*) [related]

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

Return values

uint16_t	the data value held by 't'.

12.1.2.54 DDS_ReturnCode_t CDX_DynamicType_set_boolean (CDX_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.

12.1.2.55 DDS_ReturnCode_t CDX_DynamicType_set_char (CDX_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.

12.1.2.56 DDS_ReturnCode_t CDX_DynamicType_set_default_field (CDX_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.

12.1.2.57 DDS_ReturnCode_t CDX_DynamicType_set_discriminator (CDX_DynamicType *t*, CDX_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 CDX_DynamicType_set_field_num_labels() and CDX_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.

12.1.2.58 DDS_ReturnCode_t CDX_DynamicType_set_double (CDX_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.

12.1.2.59 DDS_ReturnCode_t CDX_DynamicType_set_element (CDX_DynamicType *t,* uint32_t *n,* CDX_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.

12.1.2.60 DDS_ReturnCode_t CDX_DynamicType_set_element_type (CDX_DynamicType *t,* CDX_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.

12.1.2.61 DDS_ReturnCode_t CDX_DynamicType_set_field (CDX_DynamicType *t,* uint32_t *n,* const char * *field_name,* CDX_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.

12.1.2.62 DDS_ReturnCode_t CDX_DynamicType_set_field_label (CDX_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.

12.1.2.63 DDS_ReturnCode_t CDX_DynamicType_set_field_num_labels (CDX_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.

12.1.2.64 DDS_ReturnCode_t CDX_DynamicType_set_float (CDX_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.

12.1.2.65 DDS_ReturnCode_t CDX_DynamicType_set_length (CDX_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 CDX_DynamicType_set_element() 'n' times.

Note

If CDX_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.

12.1.2.66 DDS_ReturnCode_t CDX_DynamicType_set_long (CDX_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.

12.1.2.67 DDS_ReturnCode_t CDX_DynamicType_set_longlong (CDX_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.

12.1.2.68 DDS_ReturnCode_t CDX_DynamicType_set_max_length (CDX_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.

12.1.2.69 DDS_ReturnCode_t CDX_DynamicType_set_num_fields (CDX_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.

12.1.2.70 DDS_ReturnCode_t CDX_DynamicType_set_octet (CDX_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.

12.1.2.71 DDS_ReturnCode_t CDX_DynamicType_set_short (CDX_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.

12.1.2.72 DDS_ReturnCode_t CDX_DynamicType_set_string (CDX_DynamicType *t,* const char * *c*) [related]

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.

12.1.2.73 DDS_ReturnCode_t CDX_DynamicType_set_ulong (CDX_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.

12.1.2.74 DDS_ReturnCode_t CDX_DynamicType_set_ulonglong (CDX_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.

12.1.2.75 DDS_ReturnCode_t CDX_DynamicType_set_ushort (CDX_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.

12.2 CDX_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.

Related Functions

(Note that these are not member functions.)

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

  This operation accesses DynamicType data values (with associated DDS_SampleInfo) within the DataReader.
- DDS_ReturnCode_t CDX_DynamicTypeDataReader_take (DDS_DataReader dr, CDX_DynamicTypePtrSeq *received_data, DDS_SampleInfoSeq *sample_infos, int max_samples, DDS_SampleStateMask sample_states, DDS_ViewStateMask view_states, DDS_InstanceStateMask instance_states)

  This operation accesses DynamicType data values (with associated DDS_SampleInfo) within the DataReader.
- DDS_ReturnCode_t CDX_DynamicTypeDataReader_read_w_condition (DDS_DataReader dr, CDX_DynamicTypePtrSeq *received_data, DDS_SampleInfoSeq *sample_infos, int max_samples, DDS_ReadCondition a_condition)

  This operation accesses DynamicType data values (with associated DDS_SampleInfo) within the DataReader.
- DDS_ReturnCode_t CDX_DynamicTypeDataReader_take_w_condition (DDS_DataReader dr, CDX_DynamicTypePtrSeq *received_data, DDS_SampleInfoSeq *sample_infos, int max_samples, DDS_ReadCondition a_condition)

  This operation accesses DynamicType data values (with associated DDS_SampleInfo) within the DataReader.
- DDS_ReturnCode_t CDX_DynamicTypeDataReader_read_next_sample (DDS_DataReader dr, CDX_DynamicType received_data, DDS_SampleInfo *sample_info)

  This operation accesses DynamicType data values (with associated DDS_SampleInfo) within the DataReader.
- DDS_ReturnCode_t CDX_DynamicTypeDataReader_take_next_sample (DDS_DataReader dr, CDX_DynamicType received_data, DDS_SampleInfo *sample_info)

  This operation accessesDynamicType data values (with associated DDS_SampleInfo) within the DataReader.
- DDS_ReturnCode_t CDX_DynamicTypeDataReader_read_instance (DDS_DataReader dr, CDX_DynamicTypePtrSeq *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 DynamicType data values (with associated DDS_SampleInfo) within the DataReader.
- DDS_ReturnCode_t CDX_DynamicTypeDataReader_take_instance (DDS_DataReader dr, CDX_DynamicTypePtrSeq *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 DynamicType data values (with associated DDS_SampleInfo) within the DataReader.
- DDS_ReturnCode_t CDX_DynamicTypeDataReader_read_next_instance (DDS_DataReader dr, CDX_DynamicTypePtrSeq *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 DynamicType data values (with associated DDS_SampleInfo) within the DataReader.
- DDS_ReturnCode_t CDX_DynamicTypeDataReader_take_next_instance (DDS_DataReader dr, CDX_DynamicTypePtrSeq *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 DynamicType data values (with associated DDS_SampleInfo) within the DataReader.

- DDS_ReturnCode_t CDX_DynamicTypeDataReader_read_next_instance_w_condition (DDS_DataReader dr, CDX_DynamicTypePtrSeq ∗received_data, DDS_SampleInfoSeq ∗sample_infos, int max_samples, DDS_InstanceHandle_t previous_handle, DDS_ReadCondition a_condition)

  This operation accesses DynamicType data values (with associated DDS_SampleInfo) within the DataReader.
- DDS_ReturnCode_t CDX_DynamicTypeDataReader_take_next_instance_w_condition (DDS_DataReader dr, CDX_DynamicTypePtrSeq ∗received_data, DDS_SampleInfoSeq ∗sample_infos, int max_samples, DDS_InstanceHandle_t previous_handle, DDS_ReadCondition a_condition)

  This operation accesses DynamicType data values (with associated DDS_SampleInfo) within the DataReader.
- DDS_ReturnCode_t CDX_DynamicTypeDataReader_return_loan (DDS_DataReader dr, CDX_DynamicTypePtrSeq ∗received_data, DDS_SampleInfoSeq ∗sample_infos)

  This operation returns DynamicType data values to the DataReader.
- DDS_ReturnCode_t CDX_DynamicTypeDataReader_get_key_value (DDS_DataReader dr, CDX_DynamicType 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 CDX_DynamicTypeDataReader_lookup_instance (DDS_DataReader dr, CDX_DynamicType instance_data)

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

12.2.1 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

12.3 CDX_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.

12.3.1 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

Chapter 13

Data Structure Index

13.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 14

File Documentation

14.1 File List

Here is a list of all documented files with brief descriptions:

14.2 dds.h File Reference

CoreDX DDS C API Header File.

Data Structures

- 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.
- struct DDS_EntityFactoryQosPolicy
- struct DDS_WriterDataLifecycleQosPolicy

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

  Specifies the lifecycle behavior of data instances managed by the DataReader.
- struct CoreDX_EntityNameQosPolicy
- struct CoreDX_LoggingQosPolicy

  Controls the amount and kind of information that is logged.
- struct CoreDX_Locator

  Network address.
- struct CoreDX_ParticipantLocator

Describes a the location and identity of a potential peer DomainParticipant.

- struct CoreDX_PeerParticipantQosPolicy

  Configures a list of DomainParticipant peers to attempt communication with.
- 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 CoreDX_ThreadModelQosPolicy

  QoS Policy for configuring the threading behavior of the DomainParticipant.
- 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.

Typedefs

- typedef struct _GuardCondition * DDS_GuardCondition

  *A DDS_GuardCondition is a condition where the **trigger_value** is under application control.*
- typedef DDS_DCPSParticipant DDS_ParticipantBuiltinTopicData
- typedef struct DDS_DCPSTopic DDS_TopicBuiltinTopicData
- typedef DDS_DCPSPublication DDS_PublicationBuiltinTopicData
- typedef DDS_DCPSSubscription DDS_SubscriptionBuiltinTopicData

Enumerations

14.2.1 Detailed Description

CoreDX DDS C API Header File.

The include/dds.h file provides all of the CoreDX DDS declarations for DDS datatypes and functions. This file should be included by C application code that uses the C DDS Application Programming Interface.

14.2.2 Typedef Documentation

14.2.2.1 typedef DDS_DCPSParticipant DDS_ParticipantBuiltinTopicData

info about a discovered participant

14.2.2.2 typedef DDS_DCPSPublication DDS_PublicationBuiltinTopicData

info about a discovered publication (datawriter)

14.2.2.3 typedef DDS_DCPSSubscription DDS_SubscriptionBuiltinTopicData

info about a discovered subscription (datareader)

14.2.2.4 typedef struct DDS_DCPSTopic DDS_TopicBuiltinTopicData

info about a discovered topic

14.2.3 Enumeration Type Documentation

14.2.3.1 enum DDS_DiscoveryQosPolicyDiscoveryKind

This enumeration contains the kinds of Discovery.

Enumerator

DDS_PEER_DISCOVERY_QOS discovery kind

DDS_CENTRAL_DISCOVERY_QOS discovery kind

14.3 dds_builtin.h File Reference

Data Structures

- struct DDS_Duration_t

  The Duration_t structure contains data to define a time duration.
- struct DDS_UserDataQosPolicy

  Allows the application to attach arbitrary information to a DomainParticipant, DataWriter or DataReader QoS.
- struct DDS_TopicDataQosPolicy

  Allows the application to attach arbitrary information to a Topic QoS.
- struct DDS_GroupDataQosPolicy

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

  A hint to the middleware to help configure the transport priority mechanism.
- struct DDS_LifespanQosPolicy

  Specifies the maximum duration of validity of the data written by the DataWriter.
- struct DDS_DurabilityQosPolicy

  The DurabilityQosPolicy controls the durablity of data.
- struct DDS_PresentationQosPolicy

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

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

  Specifies allowable latency.
- struct DDS_OwnershipQosPolicy

  Determines instance ownership in the case of multple writers. CoreDX DDS supports both SHARED_OWNERSHIP_QOS and EXCLUSIVE_OWNERSHIP_QOS.
- struct DDS_OwnershipStrengthQosPolicy

  *Defines the strength, or priority, of a Writer. The strength is used to determine ownership in the case of EXCLUSIVE_OWNERSHIP_QOS. When multiple writers publish data about the same instance, the **stronger** writer is considered the owner, and data from other writers is not delivered to the reader.*
- struct DDS_LivelinessQosPolicy

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

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

  Defines a logical data partition.
- struct DDS_ReliabilityQosPolicy

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

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

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

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

- struct DDS_DurabilityServiceQosPolicy
- struct DDS_DataRepresentationQosPolicy

  Describes the data representation used by a topic.
- struct DDS_TypeConsistencyEnforcementQosPolicy

  rules for determining type consistency
- struct DDS_TypecodeQosPolicy

  Typecode representing the datatype a DataReader reads or a DataWriter writes.
- struct DDS_RpcQosPolicy

  Augment a DataWriter or DataReader with RPC specific information.
- struct DDS_Property_t

  A name-value pair property. The 'propagate' flag indicates if this property should be transfered through discovery.
- struct DDS_PropertyQosPolicy
- struct DDS_DCPSParticipant
- struct DDS_DCPSPublication
- struct DDS_DCPSSubscription

Macros

- #define DDS_VOLATILE_DURABILITY_QOS 0

  The data is volatile, and is not persisted beyond the initial publication.
- #define DDS_TRANSIENT_LOCAL_DURABILITY_QOS 1

  The data is persisted locally within the source DataWriter.
- #define DDS_TRANSIENT_DURABILITY_QOS 2
- #define DDS_PERSISTENT_DURABILITY_QOS 3
- #define DDS_INSTANCE_PRESENTATION_QOS 0

  The scope is within an Instance.
- #define DDS_TOPIC_PRESENTATION_QOS 1

  The scope is within samples published by DataWriters on the Topic.
- #define DDS_GROUP_PRESENTATION_QOS 2

  The scope is within samples published by DataWriters within the Publisher.
- #define DDS_SHARED_OWNERSHIP_QOS 0

  DataReaders receive data published by multiple DataWriters.
- #define DDS_EXCLUSIVE_OWNERSHIP_QOS 1

  DataReaders receive data published by the highest strength DataWriter for each Instance.
- #define DDS_AUTOMATIC_LIVELINESS_QOS 0

  The DomainParticipant automatically asserts liveliness for all DataWriters within the DomainParticipant.
- #define DDS_MANUAL_BY_PARTICIPANT_LIVELINESS_QOS 1

  The application manually asserts the liveliness all DataWriters within this DomainParticipant by calling assert_liveliness() or writing data on any one DataWriter within this DomainParticipant.
- #define DDS_MANUAL_BY_TOPIC_LIVELINESS_QOS 2

  The application manually asserts the liveliness for each individual DataWriter by calling assert_liveliness() or writing data on that DataWriter.
- #define DDS_BEST_EFFORT_RELIABILITY_QOS 1

  Best effort delivery – NO retransmission, HB, or ACKNACK.
- #define DDS_RELIABLE_RELIABILITY_QOS 2

  Reliable delivery – HB, ACKNACK, and re-transmission.
- #define DDS_BY_RECEPTION_TIMESTAMP_DESTINATIONORDER_QOS 0

  Use the reception time to determine the order of samples.

- #define DDS_BY_SOURCE_TIMESTAMP_DESTINATIONORDER_QOS 1

  Use the timestamp set by the publisher to determine the order of samples.
- #define DDS_KEEP_LAST_HISTORY_QOS 0

  Store at most depth Samples in the data cache for each Instance. Samples may be overwritten.
- #define DDS_KEEP_ALL_HISTORY_QOS 1

  Store Samples for each Instance (up to the limits specified by ResourceLimitsQosPolicy) until the DDS infrastructure is finished with them. Until this time, Samples may not be overwritten.
- #define DDS_XCDR_DATA_REPRESENTATION 0 /* type: DDS_DataRepresentationId_t */

  a DDS_DataRepresentationId_t that indicates XCDR form.
- #define DDS_XML_DATA_REPRESENTATION 1 /* type: DDS_DataRepresentationId_t */

  a DDS_DataRepresentationId_t that indicates XML form.
- #define DDS_DISALLOW_TYPE_COERCION 0
- #define DDS_ALLOW_TYPE_COERCION 1

Typedefs

- typedef unsigned int DDS_DurabilityQosPolicyKind

  This enumeration contains the kinds of Durability.
- typedef unsigned int DDS_PresentationQosPolicyAccessScopeKind

  This enumeration contains the kinds of Access Scope for the PresentationQosPolicy.
- typedef unsigned int DDS_OwnershipQosPolicyKind

  This enumeration contains the kinds of Ownership.
- typedef unsigned int DDS_LivelinessQosPolicyKind

  This enumeration contains the kinds of Liveliness.
- typedef unsigned int DDS_ReliabilityQosPolicyKind

  This enumeration contains the kinds of Reliability.
- typedef unsigned int DDS_DestinationOrderQosPolicyKind

  This enumeration contains the kinds of Destination Ordering.
- typedef unsigned int DDS_HistoryQosPolicyKind

  This enumeration contains the kinds of History.
- typedef short DDS_DataRepresentationId_t

  Indicates the form of data, for example CDR or XML.
- typedef unsigned short DDS_TypeConsistencyKind

  the allowed kinds of TypeConsistency

14.3.1 Macro Definition Documentation

14.3.1.1 #define DDS_ALLOW_TYPE_COERCION 1

allow coercion

14.3.1.2 #define DDS_DISALLOW_TYPE_COERCION 0

disallow coercion (type name and structure must match)

14.3.1.3 #define DDS_PERSISTENT_DURABILITY_QOS 3

The data is persisted to 'off-line' storage, and is available even after a system reboot.

14.3.1.4 #define DDS_TRANSIENT_DURABILITY_QOS 2

The data is persisted beyond the lifecycle of the originating DataWriter, and is available even after that DataWriter has been destroyed.

14.4 dds_builtin_basic.h File Reference

Data Structures

- struct DDS_BuiltinTopicKey_t
- struct DDS_GUID_t
- struct DDS_SequenceNumber_t
- struct DDS_SampleIdentity_t

Typedefs

- typedef unsigned char DDS_GuidPrefix_t[12]

14.4.1 Typedef Documentation

14.4.1.1 typedef unsigned char DDS_GuidPrefix_t[12]

Another form of builtin topic key (12 bytes).

14.5 dds_types.h File Reference

CoreDX DDS Basic Types C++ API Header File.

Data Structures

- struct DDS_Time_t
- struct DDS_SampleInfo

  The SampleInfo structure contains information associated with each sample.
- 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_QosPolicyCount

  Holds a DDS_QosPolicyId_t value and a counter to indicate the number of events associated with that PolicyId.
- 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.
- struct DDS_CacheStatistics

  Encapsulates statistics available from a DataReader or DataWriter.

Typedefs

- typedef DDS_DOMAINID_TYPE_NATIVE DDS_DomainId_t
- typedef DDS_HANDLE_TYPE_NATIVE DDS_InstanceHandle_t
- typedef int DDS_ReturnCode_t
- typedef unsigned int DDS_StatusMask

14.5.1 Detailed Description

CoreDX DDS Basic Types C++ API Header File.

14.5.2 Typedef Documentation

14.5.2.1 typedef DDS_DOMAINID_TYPE_NATIVE DDS_DomainId_t

Identifies the domain

14.5.2.2 typedef DDS_HANDLE_TYPE_NATIVE DDS_InstanceHandle_t

Identifies a data instance within the scope of a specific Entity.

14.5.2.3 typedef int DDS_ReturnCode_t

Indicates success or failure of an API operation.

14.5.2.4 typedef unsigned int DDS_StatusMask

A set of bits that select a set of statuses.

14.6 xtypes.h File Reference

Provides the X-Types DDS_DynamicDataReader and DDS_DynamicDataWriter API.

14.6.1 Detailed Description

Provides the X-Types DDS_DynamicDataReader and DDS_DynamicDataWriter API.

14.7 xtypes_dtype.h File Reference

Provides the X-Types DDS_DynamicType, DDS_DynamicData, and related APIs.

Data Structures

- struct DDS_AnnotationDescriptor

  A DDS_AnnotationDescriptor object comprises the state of an annotation as it is applied to some element.
- struct DDS_TypeDescriptor

  A DDS_TypeDescriptor comprises the state of a type.
- struct DDS_MemberDescriptor

  A DDS_MemberDescriptor comprises the state of a DDS_DynamicTypeMember.

Functions

- DDS_MAP_DECLARE (DDS_ObjectName, DDS_ObjectName, DDS_Parameters)

  A DDS_Parameters instance is a key-value map.
- DDS_MAP_DECLARE (char *, DDS_DynamicTypeMember, DDS_DynamicTypeMembersByName)

  A DDS_DynamicTypeMembersByName instance is a map of DDS_DynamicTypeMember's indexed by 'name'.
- DDS_MAP_DECLARE (DDS_MemberId, DDS_DynamicTypeMember, DDS_DynamicTypeMembersById)

  A DDS_DynamicTypeMembersByName instance is a map of DDS_DynamicTypeMember's indexed by MemberId.
- DECLARE_SEQ (DDS_DynamicTypeMember, DDS_DynamicTypeMemberSeq)

  A DDS_DynamicTypeMemberSeq instance is a sequence of DDS_DynamicTypeMember's.

14.7.1 Detailed Description

Provides the X-Types DDS_DynamicType, DDS_DynamicData, and related APIs.

14.8 coredx_lmt_transport.h File Reference

CoreDX DDS Local Machine Transport (LMT)

Data Structures

- struct CoreDX_LmtTransportConfig

  Structure that holds LMT Transport configuration items.

14.8.1 Detailed Description

CoreDX DDS Local Machine Transport (LMT)

14.9 coredx_tcp_transport.h File Reference

CoreDX DDS TCP Transport.

Data Structures

- struct CoreDX_TcpTransportConfig

  Structure that holds TCP Transport configuration items.

14.9.1 Detailed Description

CoreDX DDS TCP Transport.

14.10 coredx_udp_transport.h File Reference

CoreDX DDS UDP Transport (default)

Data Structures

- struct CoreDX_UdpTransportConfig

  Structure that holds UDP Transport configuration items.

14.10.1 Detailed Description

CoreDX DDS UDP Transport (default)

14.11 dds_dtype.h File Reference

CoreDX DDS C DynamicType API Header File.

14.11.1 Detailed Description

CoreDX DDS C DynamicType API Header File.

The include/dds_dtype.h file provides all of the CoreDX DDS declarations for CoreDX DDS DynamicTypes. This file should be included by C application code that uses the CoreDX DDS DynamicType Application Programming Interface.

Deprecated NOTE: This API is deprecated. The preferred DynamicType API is provided in xtypes_dtype.h

14.12 dds_dtype_dr.h File Reference

14.13 dds_dtype_dw.h File Reference

14.14 dds_typecode.h File Reference

Enumerations

Chapter 15

Not Yet Supported

Global DDS_AnnotationDescriptor_copy_from
This operation is not yet implemented.

Global DDS_AnnotationDescriptor_equals
This operation is not yet implemented.

Global DDS_AnnotationDescriptor_get_all_value
This operation is not yet implemented.

Global DDS_AnnotationDescriptor_get_type
This operation is not yet implemented.

Global DDS_AnnotationDescriptor_get_value
This operation is not yet implemented.

Global DDS_AnnotationDescriptor_is_consistent
This operation is not yet implemented.

Global DDS_AnnotationDescriptor_set_value
This operation is not yet implemented.

Global DDS_DomainParticipant_create_multitopic
This operation is not implemented yet.

Global DDS_DomainParticipant_delete_multitopic
This operation is not implemented yet.

Global DDS_DomainParticipant_get_discovered_topic_data

Global DDS_DomainParticipant_ignore_publication
This operation is not implemented yet.

Global DDS_DomainParticipant_ignore_subscription
This operation is not implemented yet.

Global DDS_DomainParticipant_ignore_topic
This operation is not implemented yet.

Global DDS_DynamicType_get_annotation
This operation is not yet implemented.

Global DDS_DynamicType_get_annotation_count

This operation is not yet implemented.

Global DDS_DynamicTypeBuilder_apply_annotation

This operation is not yet implemented.

Global DDS_DynamicTypeBuilder_get_annotation

This operation is not yet implemented.

Global DDS_DynamicTypeBuilder_get_annotation_count

This operation is not yet implemented.

Global DDS_DynamicTypeBuilderFactory_create_type_w_document

This operation is not yet implemented.

Global DDS_DynamicTypeBuilderFactory_create_type_w_uri

This operation is not yet implemented.

Global DDS_DynamicTypeMember_get_annotation

This operation is not yet implemented.

Global DDS_DynamicTypeMember_get_annotation_count

This operation is not yet implemented.

Class DDS_MultiTopic

CoreDX does not yet implement MultiTopics.

Global DDS_Publisher_begin_coherent_changes

This operation is not yet implemented.

Global DDS_Publisher_end_coherent_changes

This operation is not yet implemented.

Global DDS_Publisher_resume_publications

This operation is not yet implemented.

Global DDS_Publisher_suspend_publications

This operation is not yet implemented.

Class DDS_QueryCondition

CoreDX DDS does not yet implement QueryConditions as a trigger for a WaitSet.

Chapter 16

Deprecated List

Global CoreDX_DDS_get_lib_version

Use CoreDX_DDS_get_lib_version_string() instead.

File dds_dtype.h

NOTE: This API is deprecated. The preferred DynamicType API is provided in xtypes_dtype.h

Chapter 17

Brief Type List

17.1 Data Structures

Here are the data structures with brief descriptions:

Index