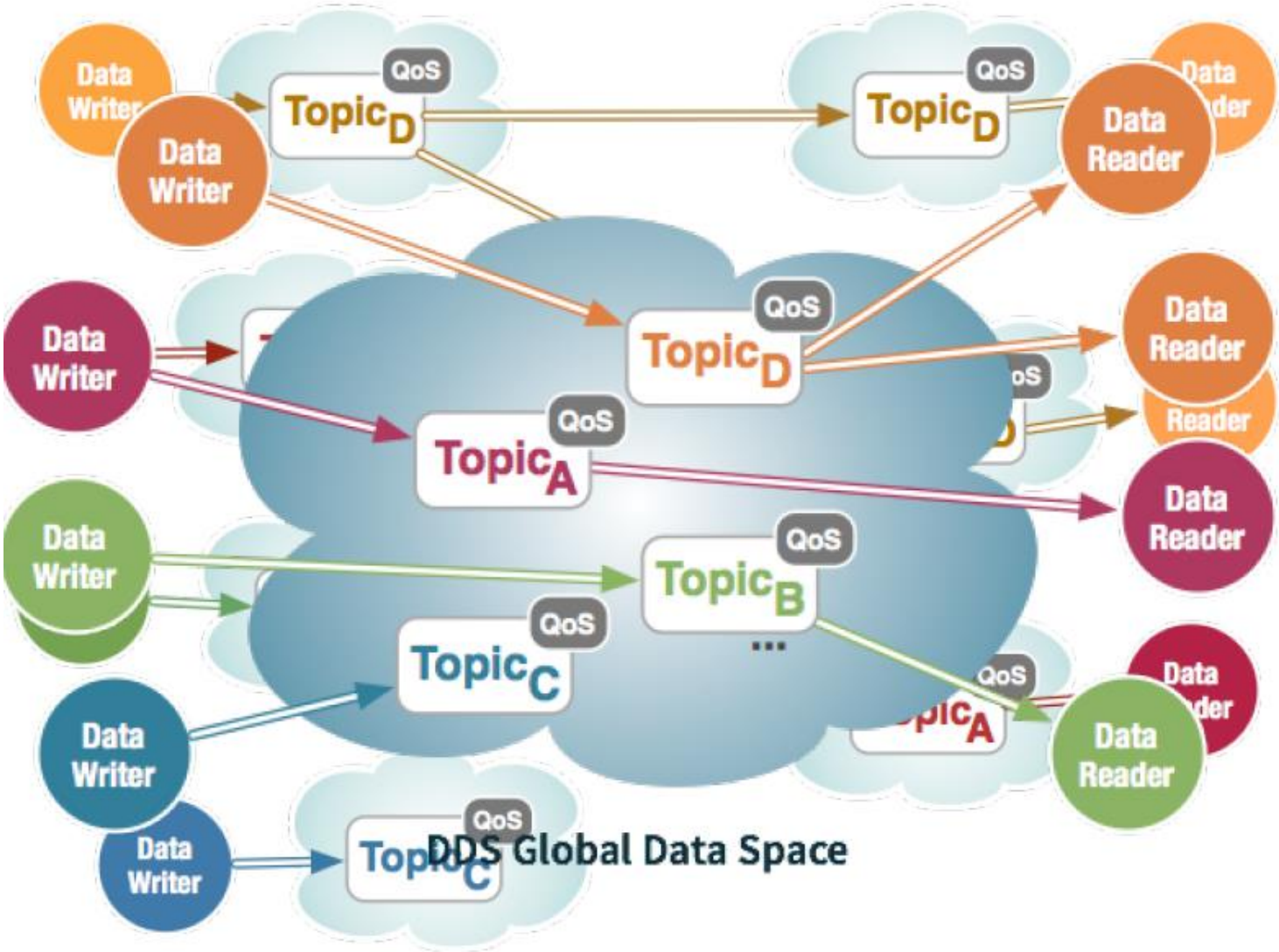
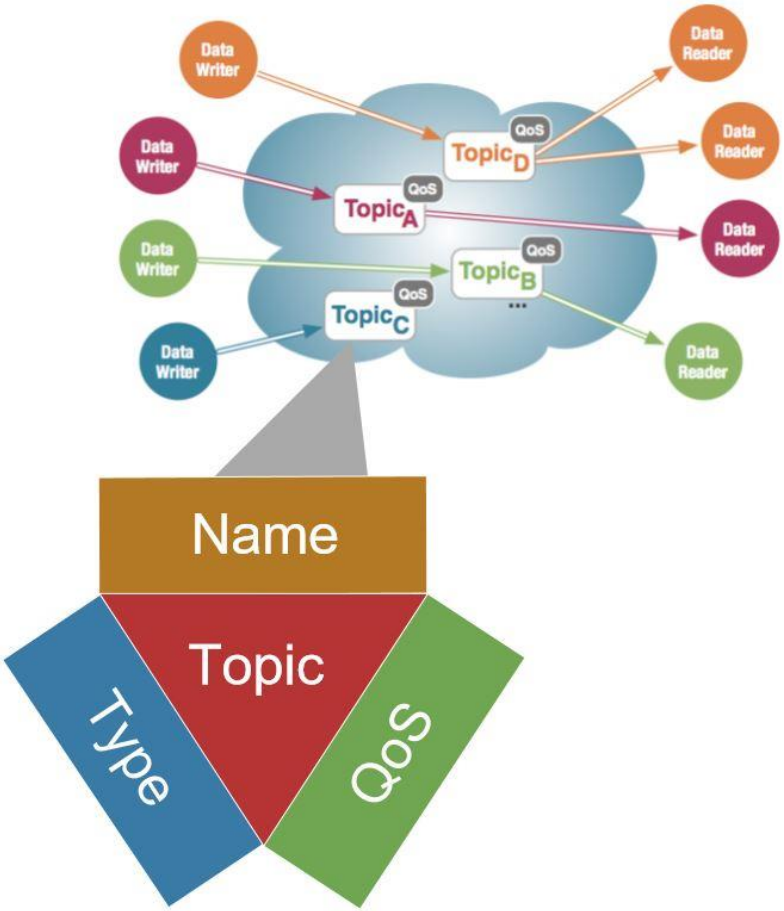
OMG DATA DISTRIBUTION SYSTEM FOR REAL-TIME SYSTEMS

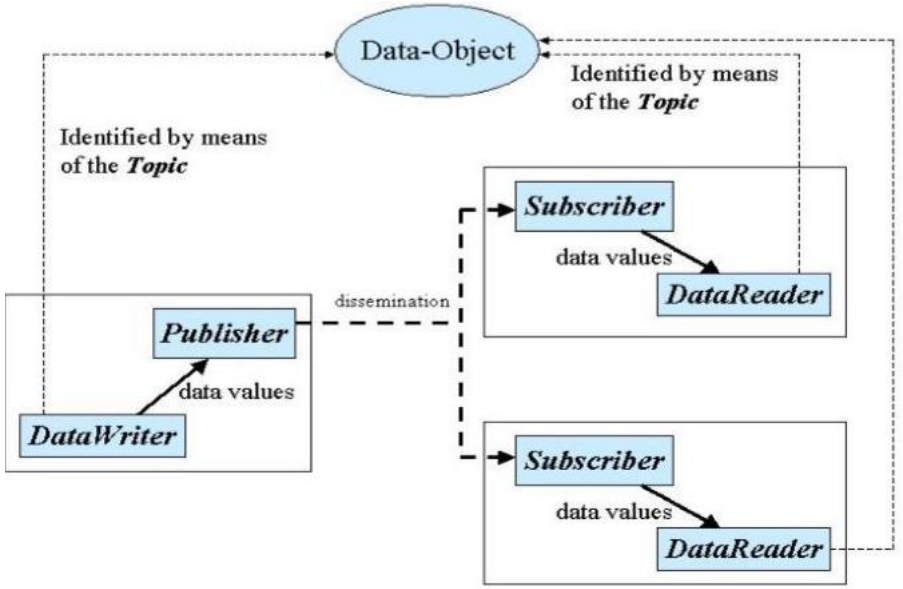
Scalability → Flexibility → Publish-subscribe
Real-time → high-performane (latency: 25-30 µsec, throughput>10G…)
Interoperability
Platform independent,
Polyglot (Ada, C, C++, C#, .Net, Java, JavaScript, Scala, Lua, Ruby)

Decentralised Data-Space

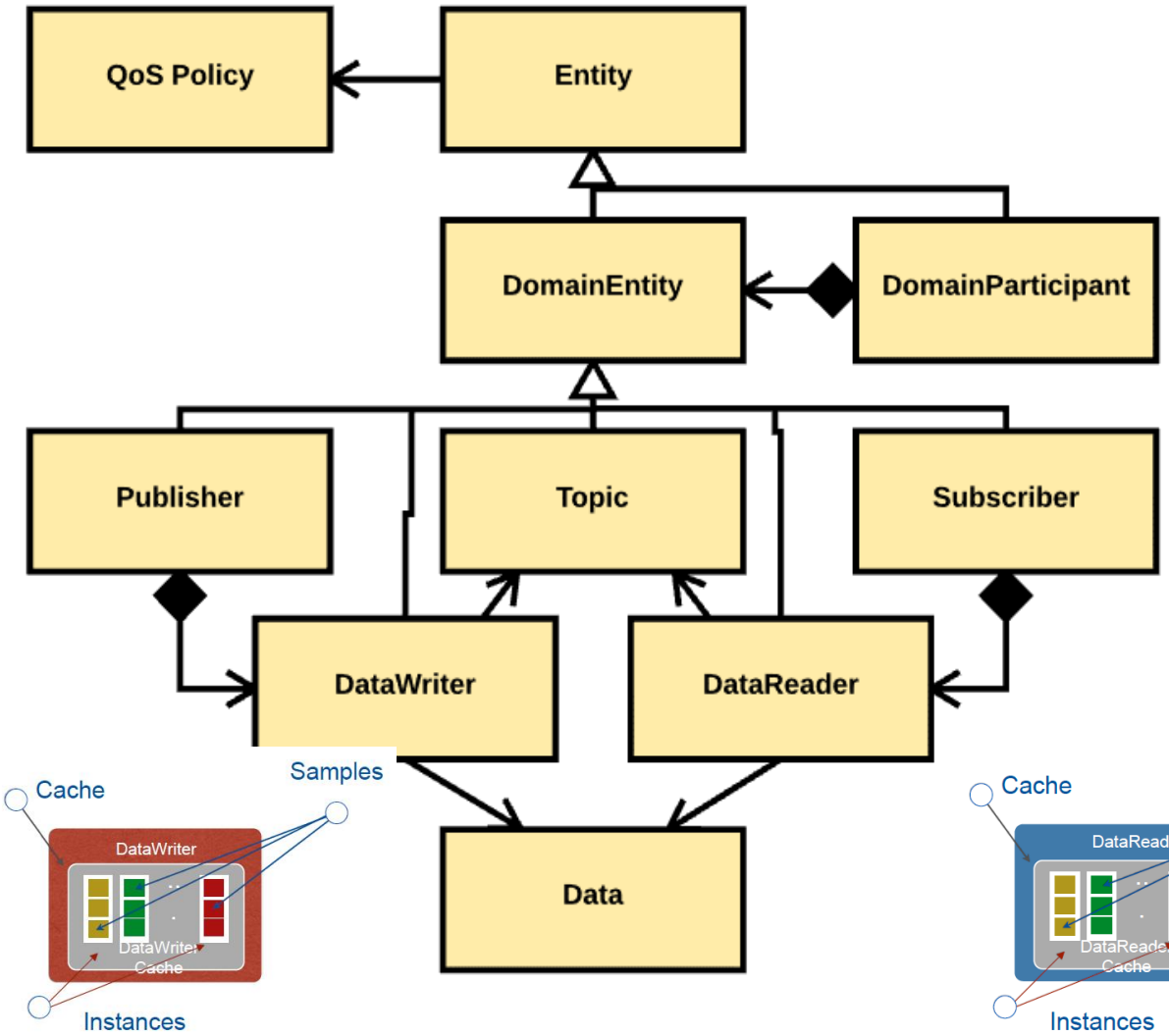


OMG DDS Core notions





DDS notions


QoS Policy
Entity
DomainEntity
DomainParticipant
Publisher
Topic
Subscriber
DataWriter
DataReader
Data
Cache
Samples
DataWriter
DataWriter Cache
Instances
Cache
Samples
DataReader
DataReader Cache
Instances

RTI Connext DDS

- Professional
  - https://www.rti.com/products/connext-dds-professional
- Secure
  - https://www.rti.c[illegible]-secure
- Micro
  - https://www.rti.[illegible]-dds-micro
- Cert
  - https:/[illegible]ts/connext-dds-cert

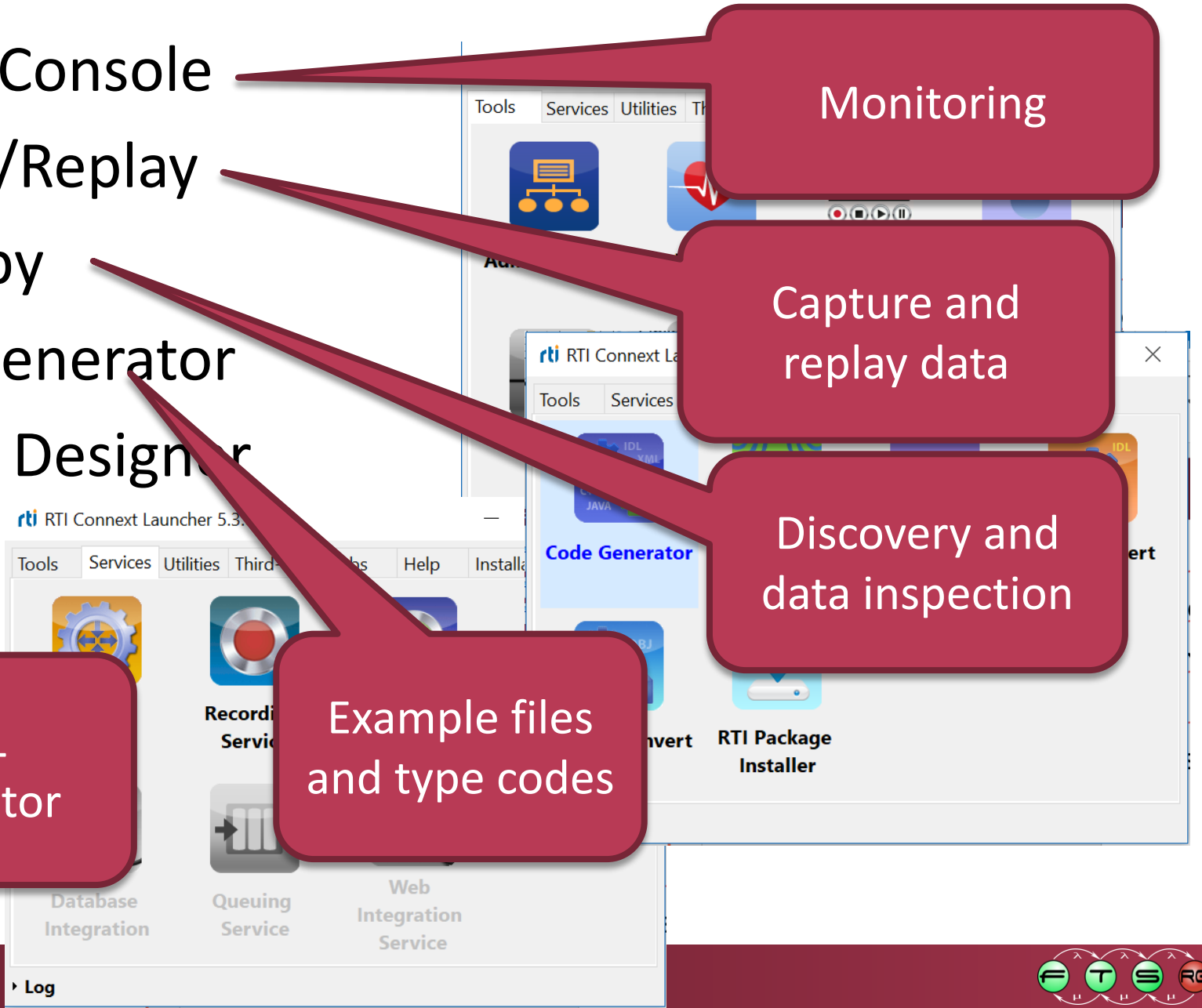
DDS security standard, architecture and model, topic level protection

Small-footprint, limited resources, µC

Safety-critical IIot,

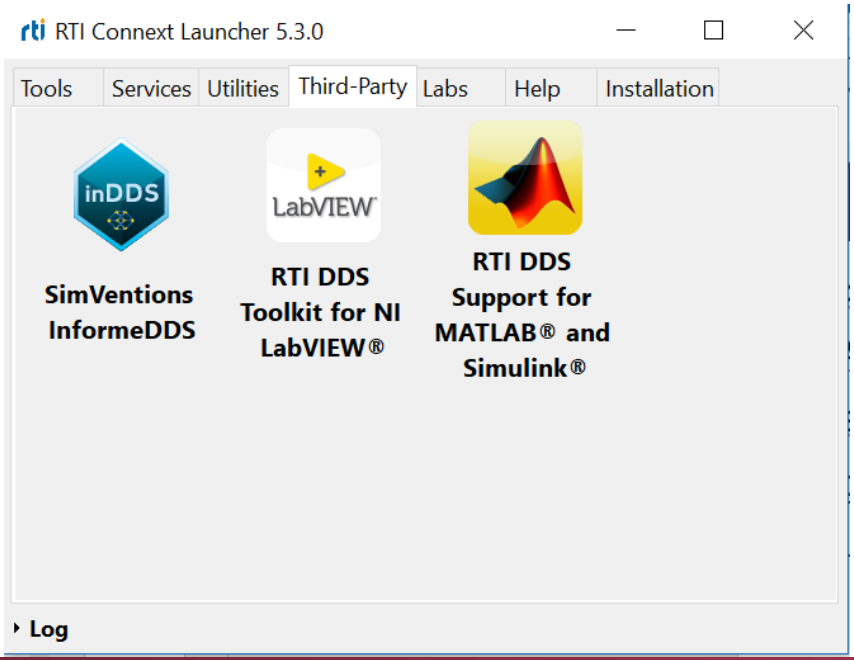
RTI Tools

- Admin Console
- Record/Replay
- Ping/Spy
- Code Generator
- System Designer



RTI 3rd Party Integration

- LabVIEW
- MATLAB,Simulink
- MagicDraw – SysML
- https://www.rti.com/products/third-party-integrations



LET'S WRITE CODE

Subscriber, Publisher, …

Copy Some Files

- HelloSubscriber.java

  Code frame of listening application

  - data reader

- HelloPublisher.java

  Code frame of speaking application

  - data writer

- USER_QOS_PROFILES.xml

  Some QoS settings to play with

  - reliable communication (with history)
  - domain partitions

Create a Subscriber

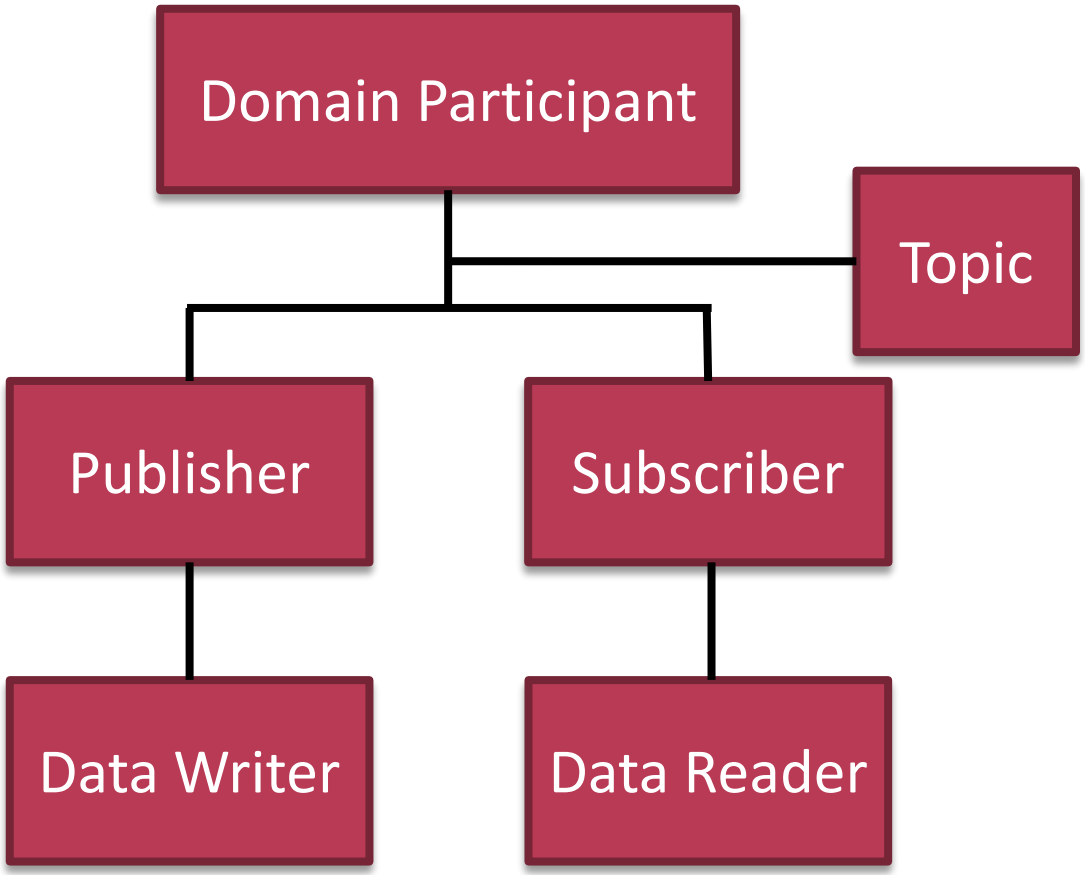
- In Java
- In Eclipse
  - create a Java project: Demo1
  - with one package in it: Subscriber
  - with one class in it: HelloSubscriber.java

Frame

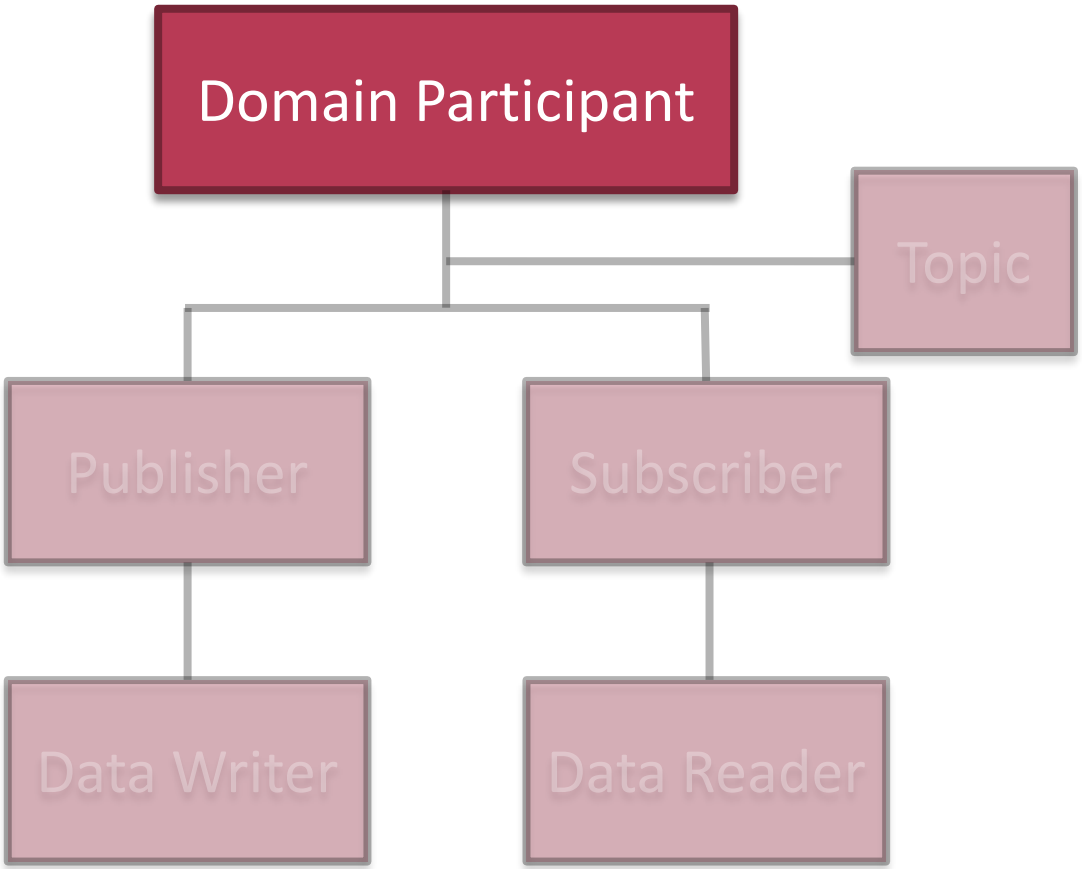
```
package Subscriber;

import com.rti.dds.subscription.*;
import com.rti.dds.domain.*;
import com.rti.dds.infrastructure.*;
import com.rti.dds.topic.*;
import com.rti.dds.type.builtin.*;

public class HelloSubscriber extends DataReaderAdapter {

…

}
```

(project → properties → Java Build Path → Libraries → Add External JARs … → C:\Program Files\rti_connext_dds-5.3.1\lib\java\nddsjava.jar)

DDS Anatomy


Domain Participant
Topic
Publisher
Subscriber
Data Writer
Data Reader

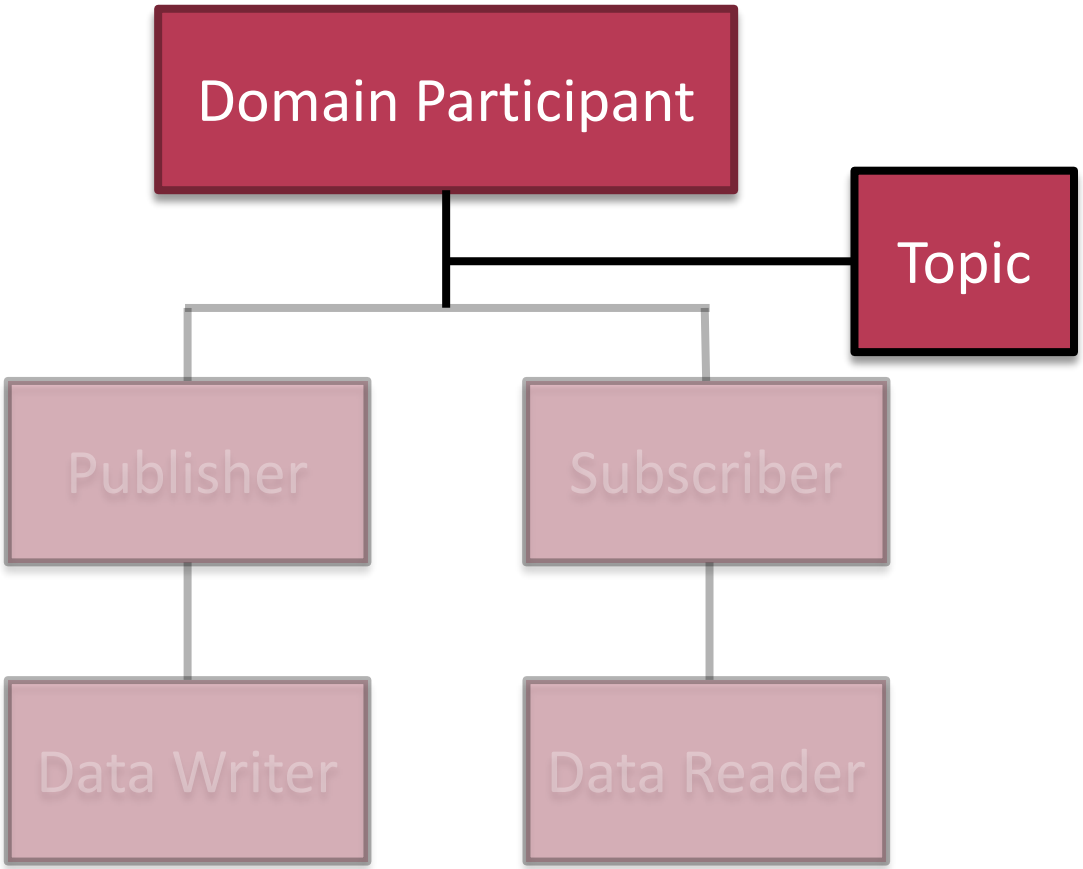
DDS Anatomy


Domain Participant
Topic
Publisher
Subscriber
Data Writer
Data Reader

DomainParticipant

```
public class HelloSubscriber extends DataReaderAdapter {

// For clean shutdown sequence
private static boolean shutdown_flag = false;

public static final void main(String[] args) {
        DomainParticipant participant =
        DomainParticipantFactory.get_instance().create_participant(
                0,                                          // Domain ID = 0
                DomainParticipantFactory.PARTICIPANT_QOS_DEFAULT, // QoS
                null,                                       // Listener
                StatusKind.STATUS_MASK_NONE);               // Mask

if (participant == null) { System.err.println("Unable to create domain participant");
                           return; }
```

DDS Anatomy


Domain Participant
Topic
Publisher
Subscriber
Data Writer
Data Reader

Topic

```
Topic topic = participant.create_topic(
                „World temperature",                    // Topic Name
                StringTypeSupport.get_type_name(),      // Topic Data Type
                DomainParticipant.TOPIC_QOS_DEFAULT,    // QoS
                null,                                   // Listener
                StatusKind.STATUS_MASK_NONE);           // Mask

if (topic == null) { System.err.println("Unable to create topic.");
                     return; }
```

Data Structure Definition (IDL)

```
const long MAX_BUFFER_SIZE = 1048576;

struct VideoStream {

        // This allows a subscriber to differentiate between different video
        // publishers that are publishing the same data on the same Topic.
        long stream_id; //@key

        // Some video formats may require metadata to be sent with the binary
        // video data, such as flags or a frame sequence_number.
        unsigned long flags;
        unsigned long sequence_number;

        // This contains the video buffers
        sequence <octet, MAX_BUFFER_SIZE> frame;
};
```

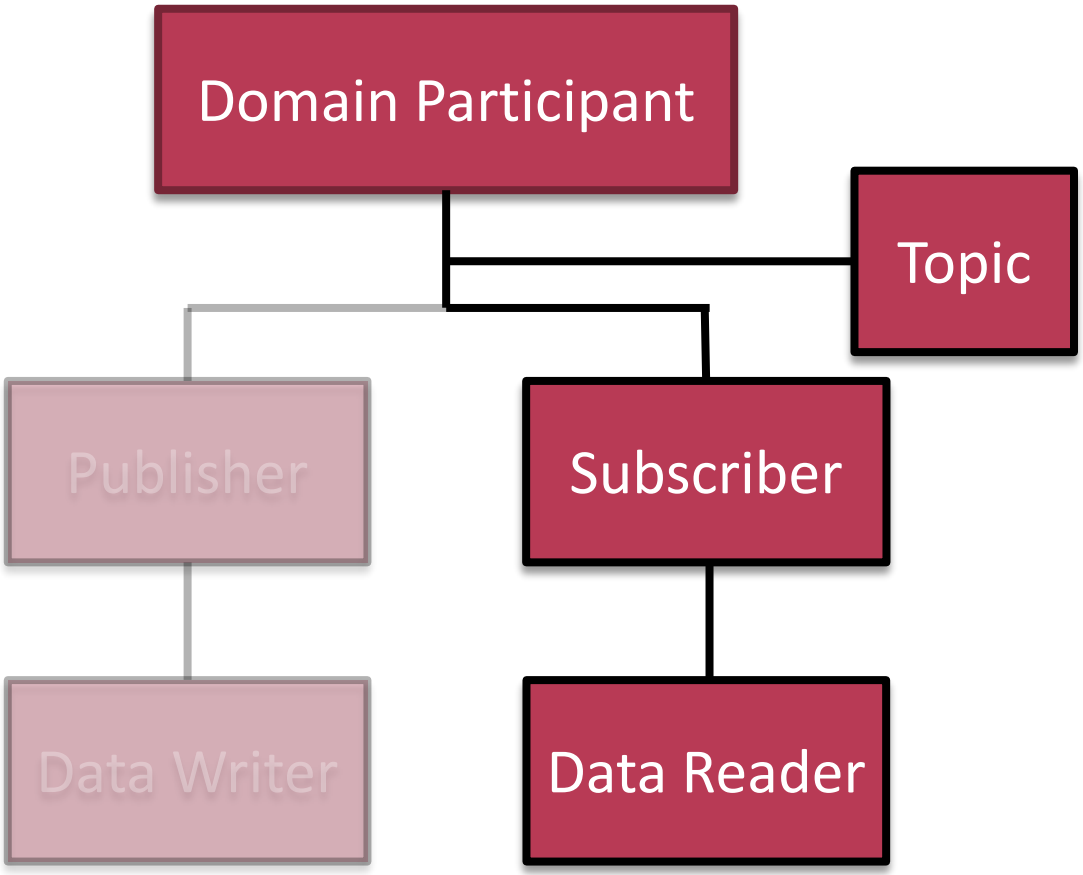
Data Structure Definition (IDL)

```
const long MAX_BUFFER_SIZE

struct VideoStream {

        // This allows a subscrib
        // publishers that are pub
        long stream_id; //@key

        // Some video formats may require metadata to be sent with the binary
        // video data, such as flags or a frame sequence_number.
        unsigned long flags;
        unsigned long sequence_number;

        // This contains the video buffers
        sequence <octet, MAX_BUFFER_SIZE> frame;
};
```

rtiddsgen⇒

- VideoStream.java
- VideoStreamDataReader.java
- VideoStreamDataWriter.java
- VideoStreamSeq.java
- VideoStreamTypeCode.java
- VideoStreamTypeSupport.java

DDS Anatomy


Domain Participant
Topic
Publisher
Subscriber
Data Writer
Data Reader

DataReader

```
StringDataReader dataReader = (StringDataReader) participant.create_datareader(
        topic,                                  //Topic
        Subscriber.DATAREADER_QOS_DEFAULT,      // QoS
        new HelloSubscriber(),                  // Listener
        StatusKind.DATA_AVAILABLE_STATUS);      // Mask

if (dataReader == null) { System.err.println("Unable to create DDS Data Reader");
                          return; }

//Reading User-Input
System.out.println("Ready to read data.");
System.out.println("Press CTRL+C to terminate.");
```

Could also be a filtered topic.

Shutdown

```
for (;;) {
        try {
                Thread.sleep(2000);
                 if(shutdown_flag) break; }
        catch (InterruptedException e) {
                // Nothing to do... }
}

System.out.println("Shutting down...");

//Deleting entities and DomainParticipant
participant.delete_contained_entities();
DomainParticipantFactory.get_instance().delete_participant(participant);
}
```

Listener

```
public void on_data_available(DataReader reader) {

        StringDataReader stringReader = (StringDataReader) reader;
        SampleInfo info = new SampleInfo();

        for (;;) {  try {
                        String sample = stringReader.take_next_sample(info);
                        if (info.valid_data) {
                                System.out.println(sample);
                                if (sample.equals("")) { shutdown_flag = true;}
                        }
                }
                catch (RETCODE_NO_DATA noData) { break; }
                catch (RETCODE_ERROR e) { e.printStackTrace(); }
        }
}
```

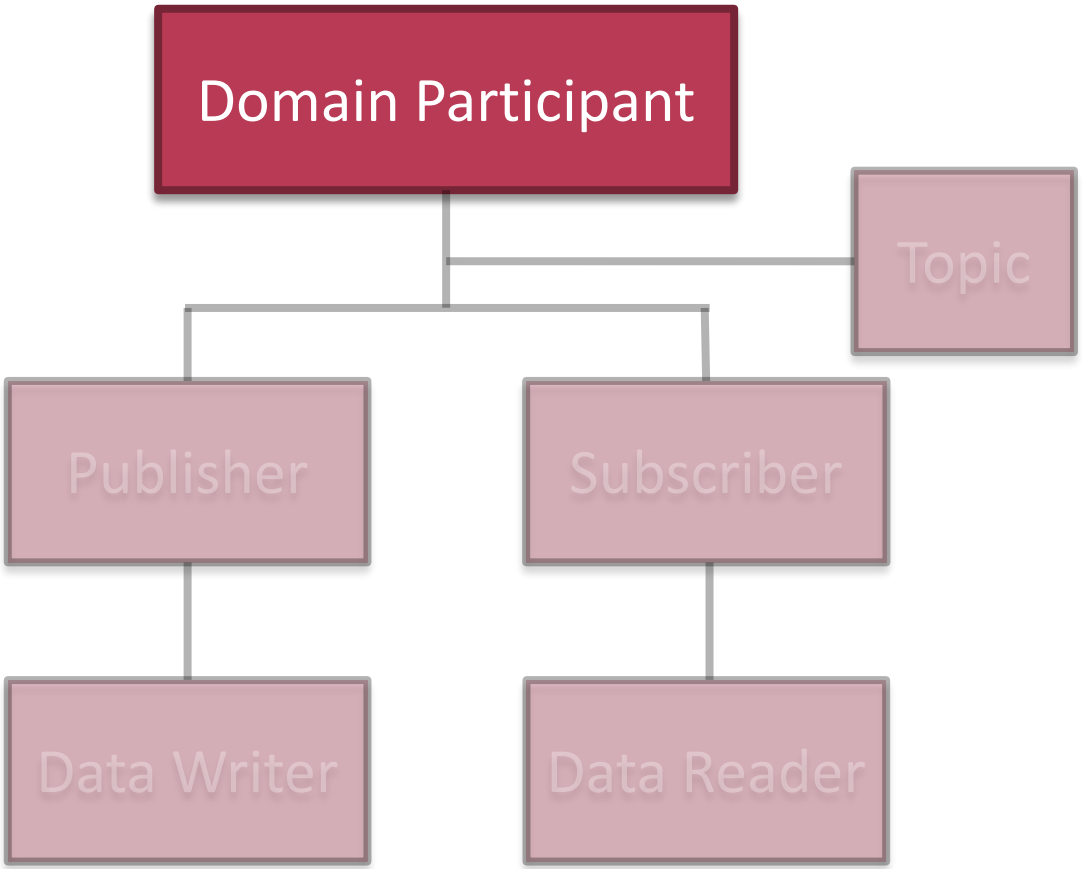
Create a Publisher

- In Java
- In Eclipse
  - you have a Java project: Demo1
  - add one package in it: Publisher
  - with one class in it: HelloPublisher.java

Frame

```
package Publisher;
import java.io.*;

import com.rti.dds.publication.*;
import com.rti.dds.domain.*;
import com.rti.dds.infrastructure.*;
import com.rti.dds.topic.*;
import com.rti.dds.type.builtin.*;

public class HelloPublisher {

...

}
```

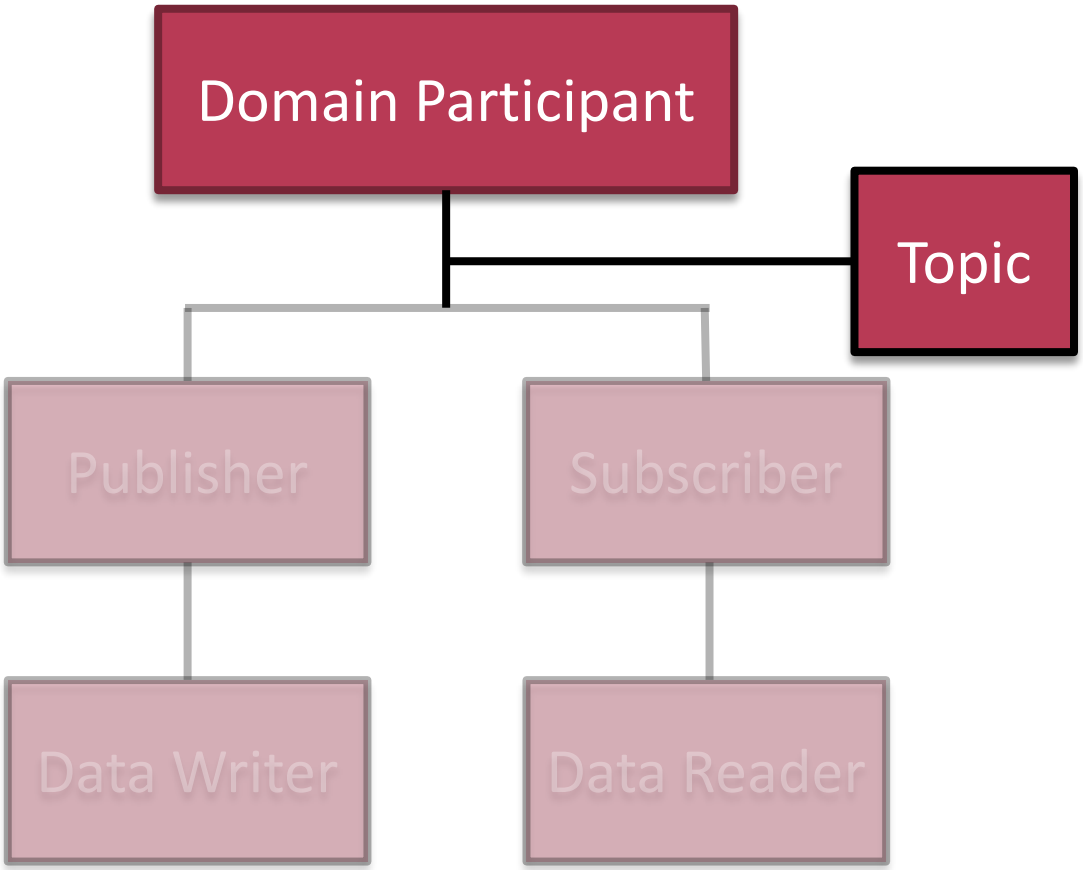
DDS Anatomy


Domain Participant
Topic
Publisher
Subscriber
Data Writer
Data Reader

DomainParticipant

```
public class HelloPublisher {

public static final void main(String[] args) {
        DomainParticipant participant =
        DomainParticipantFactory.get_instance().create_participant(
                0,                                          // Domain ID = 0
                DomainParticipantFactory.PARTICIPANT_QOS_DEFAULT, // QoS
                null,                                       // Listener
                StatusKind.STATUS_MASK_NONE);               // Mask

        if (participant == null) {
                System.err.println("Unable to create domain participant");
                return;
        }
```

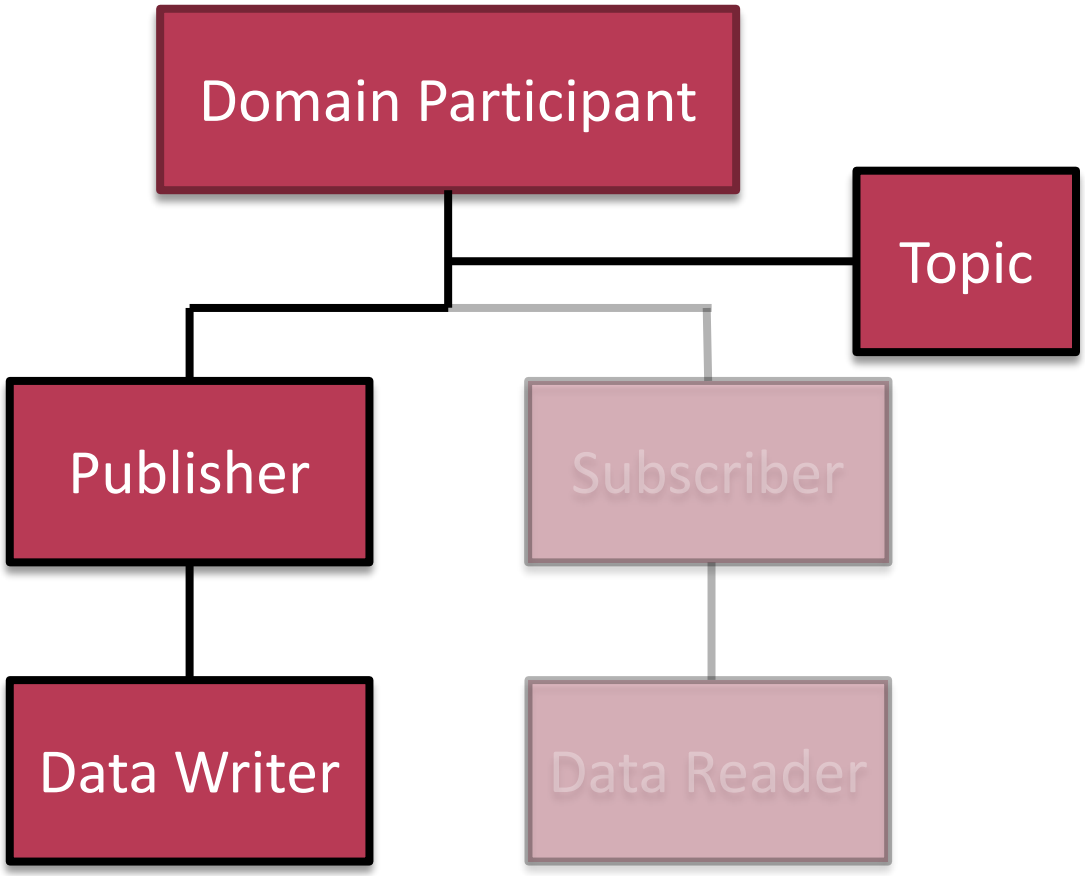
DDS Anatomy


Domain Participant
Topic
Publisher
Subscriber
Data Writer
Data Reader

Topic

```
Topic topic = participant.create_topic(
                „ World temperature",                    // Topic Name
                StringTypeSupport.get_type_name(),       // Topic Data Type
                DomainParticipant.TOPIC_QOS_DEFAULT,     // QoS
                null,                                    // Listener
                StatusKind.STATUS_MASK_NONE);            // Mask

if (topic == null) { System.err.println("Unable to create topic.");
                     return; }
```

DDS Anatomy


Domain Participant
Topic
Publisher
Subscriber
Data Writer
Data Reader

DataWriter

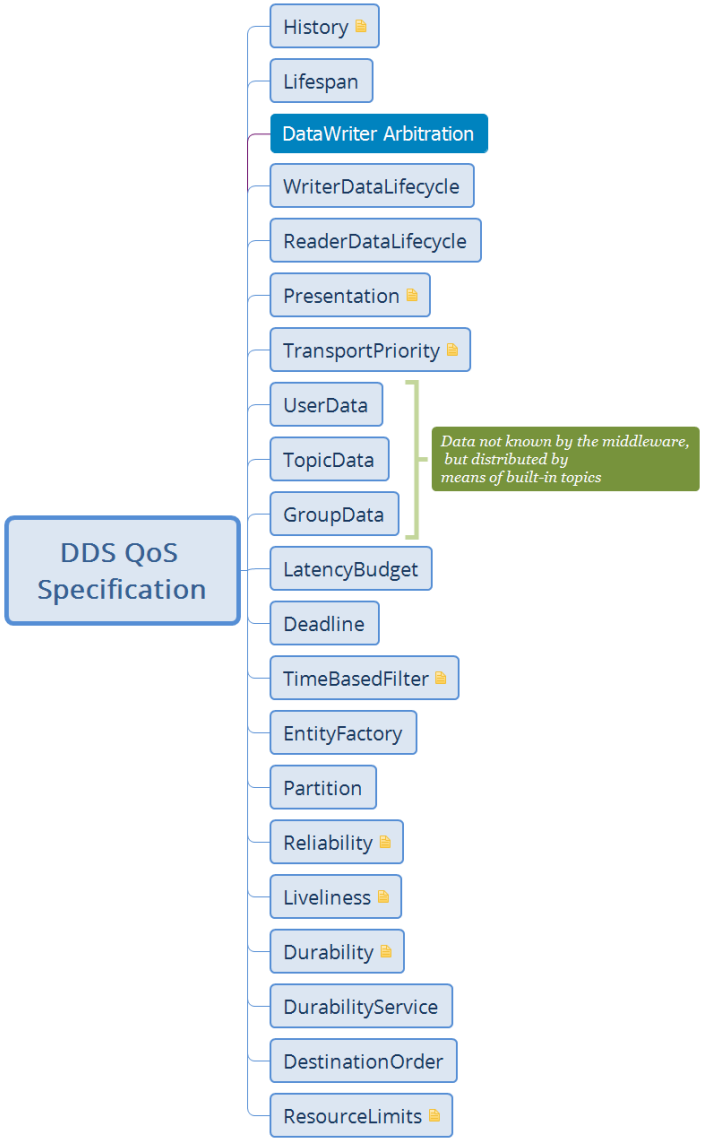
```
StringDataWriter dataWriter = (StringDataWriter) participant.create_datawriter(
                    topic,                                  //Topic
                    Publisher.DATAWRITER_QOS_DEFAULT,       // QoS
                    null,                                   // Listener
                    StatusKind.DATA_AVAILABLE_STATUS);      // Mask

if (dataWriter == null) { System.err.println("Unable to create DDS Data Writer");
                          return; }

System.out.println("Ready to write data.");
System.out.println("When the subscriber is ready, you can start writing.");
System.out.print("Press CTRL+C to terminate or enter an empty line to do a clean
shutdown.\n\n");
```

Read from User

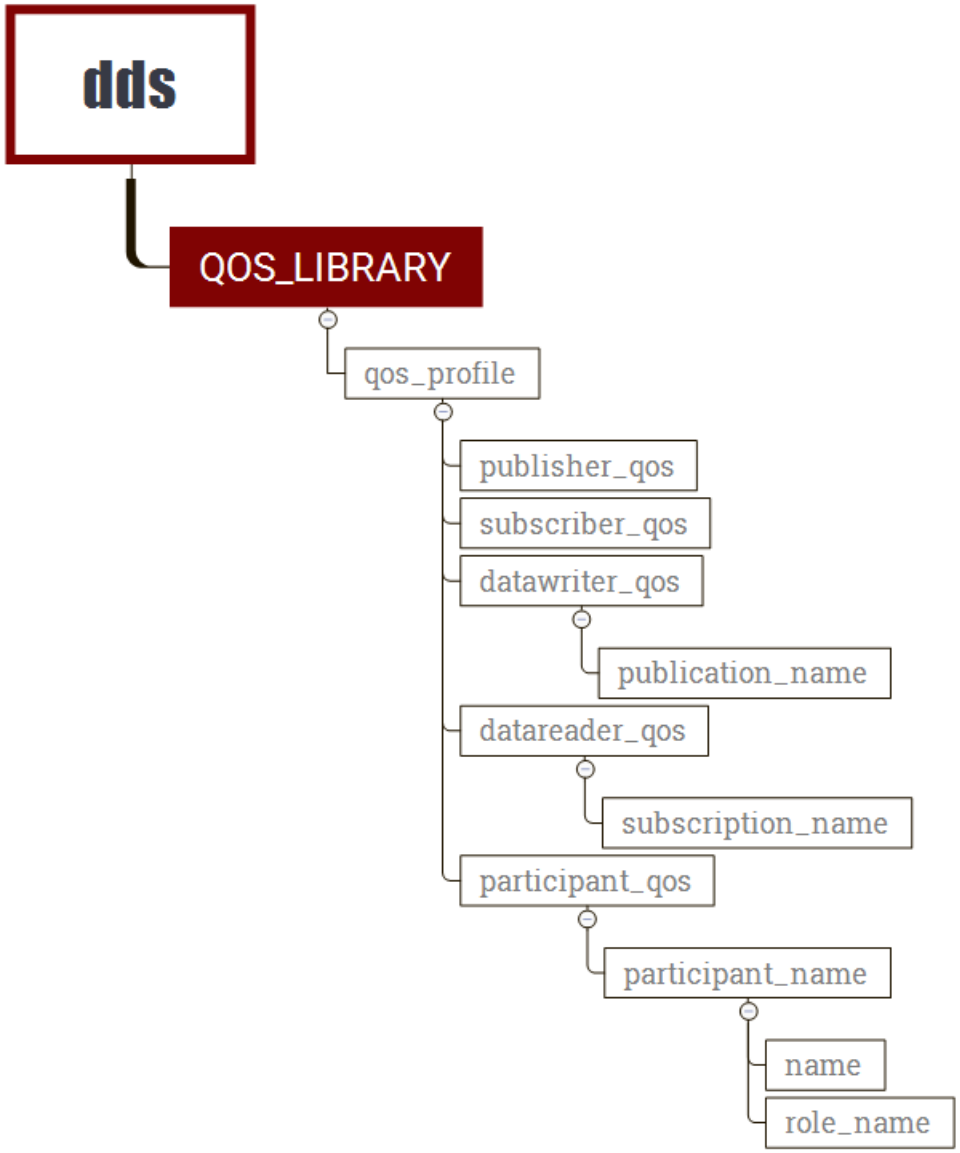
```
BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));
try { while (true) {
                    System.out.print("Please type a message> ");
                    String toWrite = reader.readLine();
                    dataWriter.write(toWrite, InstanceHandle_t.HANDLE_NIL);
                    if (toWrite.equals("")) break;
                    }
}
catch (IOException e) { e.printStackTrace(); }
catch (RETCODE_ERROR e) { e.printStackTrace(); }

System.out.println("Exiting...");
participant.delete_contained_entities();
DomainParticipantFactory.get_instance().delete_participant(participant);
}
```

LET’S PLAY AROUND WITH QOS

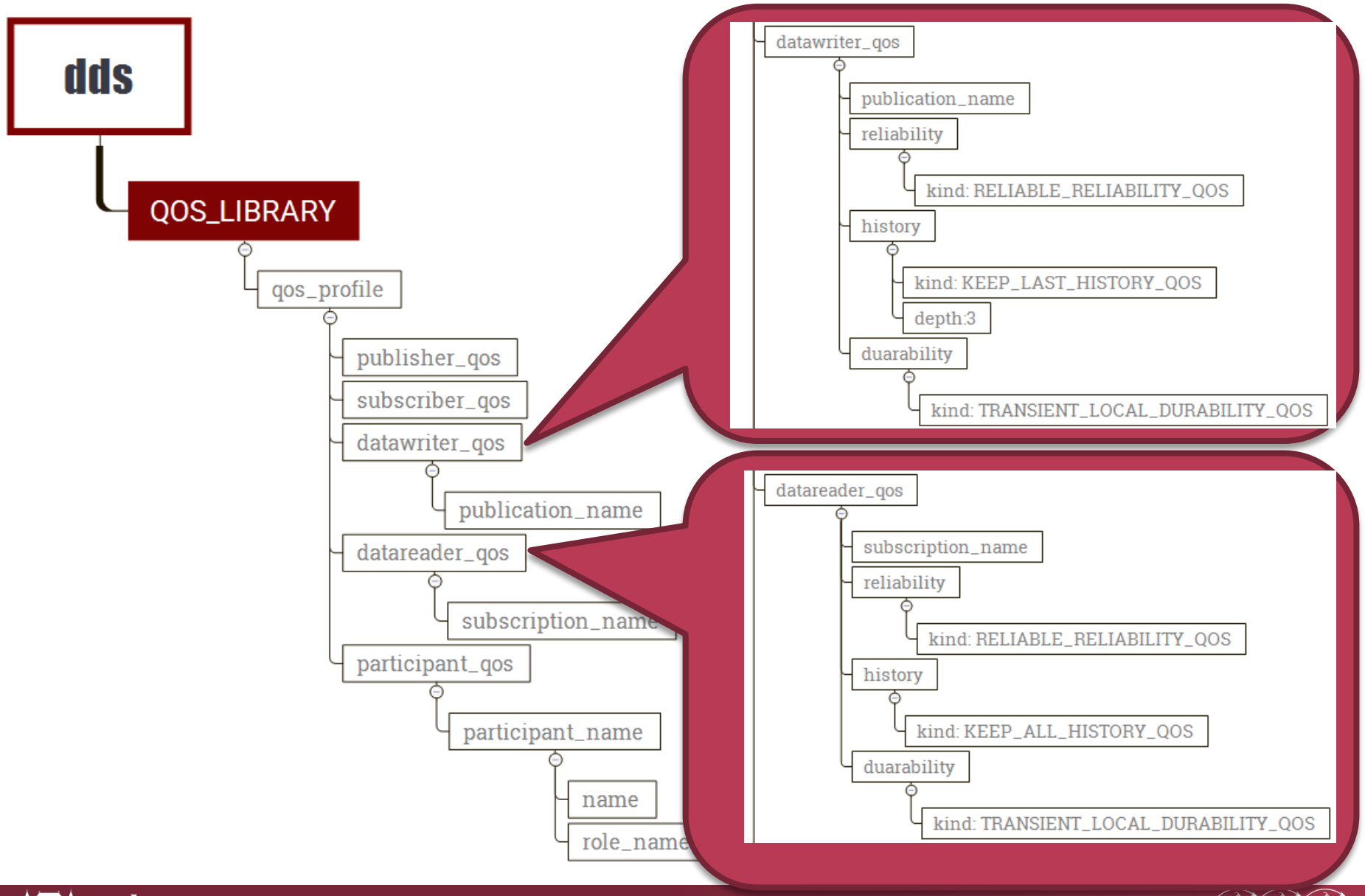
DDS QoS


DDS QoS Specification
History
Lifespan
DataWriter Arbitration
WriterDataLifecycle
ReaderDataLifecycle
Presentation
TransportPriority
UserData
TopicData
GroupData
Data not known by the middleware, but distributed by means of built-in topics
LatencyBudget
Deadline
TimeBasedFilter
EntityFactory
Partition
Reliability
Liveliness
Durability
DurabilityService
DestinationOrder
ResourceLimits

USER_QOS_PROFILES.XML



Reliability (XML)



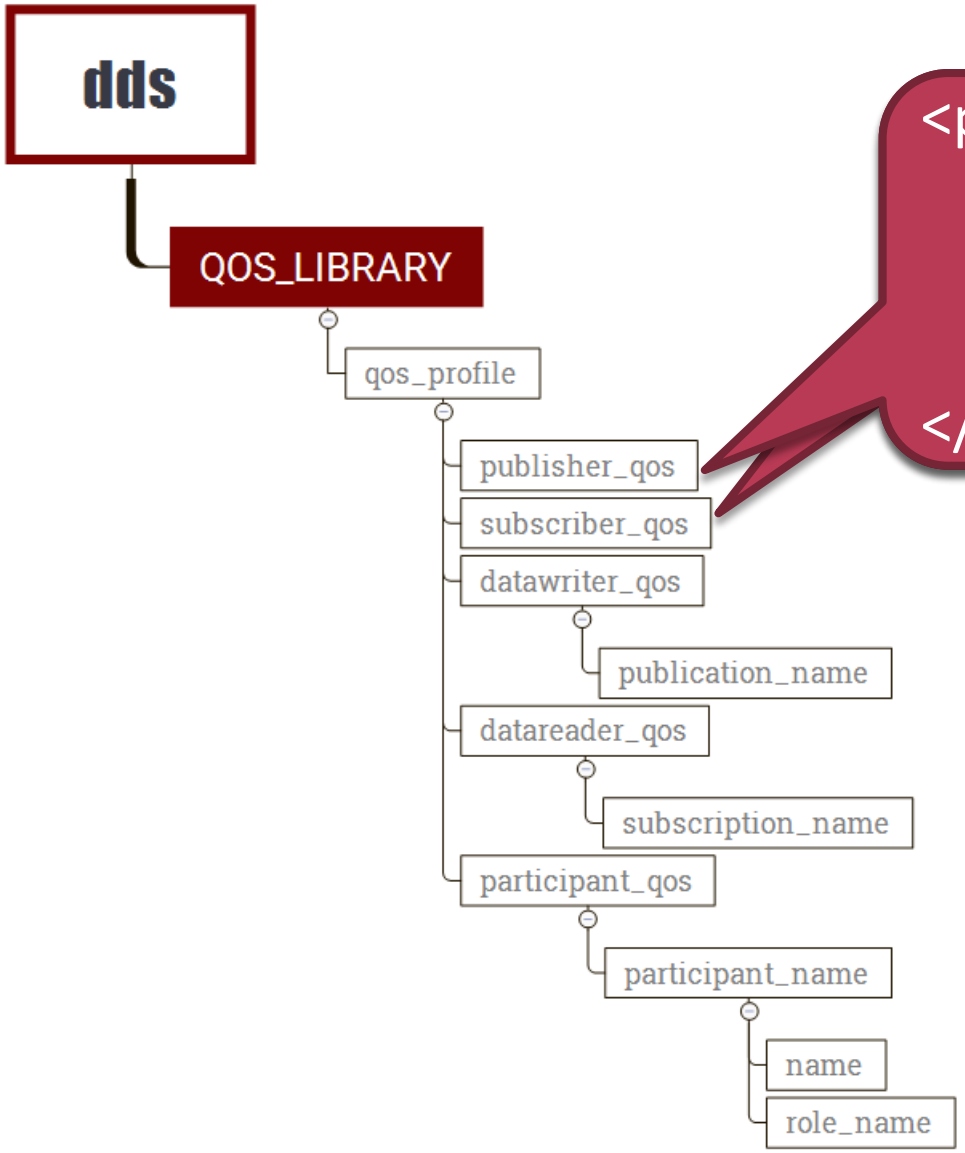
Reliability (Java)

```
DataWriterQos writer_qos = new DataWriterQos();
participant.get_default_datawriter_qos(writer_qos);

writer_qos.reliability.kind =ReliabilityQosPolicyKind.RELIABLE_RELIABILITY_QOS;
writer_qos.history.kind = HistoryQosPolicyKind.KEEP_LAST_HISTORY_QOS;
writer_qos.history.depth = 3;
writer_qos.durability.kind =
                    DurabilityQosPolicyKind.TRANSIENT_LOCAL_DURABILITY_QOS;

// Create the data writer using the default publisher
StringDataWriter dataWriter = (StringDataWriter) participant.create_datawriter(
                    topic,                                  //Topic
                    writer_qos,                             // QoS
                    null,                                   // Listener
                    StatusKind.DATA_AVAILABLE_STATUS);      // mask
```

USER_QOS_PROFILES.xml (→Demo1 könyvtár)

```
<?xml version="1.0"?>
<dds xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance xsi:noNamespaceSchemaLocation="C:/Program Files/rti_connext_dds-5.3.0/resource/schema/rti_dds_qos_profiles.xsd" version="5.3.0">
<qos_library name="Demo1_Library">
    <qos_profile name="Demo1_Profile" base_name="BuiltinQosLibExp::Generic.StrictReliable" is_default_qos="true">
        <!-- QoS used to configure the data writer created in the example code -->
        <publisher_qos><partition><name><element>A*C</element></name></partition></publisher_qos>
        <subscriber_qos><partition><name><element>AB*</element></name></partition></subscriber_qos>
        <datawriter_qos>
                <publication_name><name>HelloPublisher</name></publication_name>
                <reliability><kind>RELIABLE_RELIABILITY_QOS</kind></reliability>
                <history><kind>KEEP_LAST_HISTORY_QOS</kind><depth>3</depth></history>
                <durability><kind>TRANSIENT_LOCAL_DURABILITY_QOS</kind></durability>
        </datawriter_qos>
        <datareader_qos>
                <subscription_name><name>HelloSubscriber</name></subscription_name>
                <reliability><kind>RELIABLE_RELIABILITY_QOS</kind></reliability>
                <history><kind>KEEP_ALL_HISTORY_QOS</kind></history>
                <durability><kind>TRANSIENT_LOCAL_DURABILITY_QOS</kind></durability>
        </datareader_qos>
        <participant_qos><participant_name><name>Demo1Participant</name><role_name>Demo1ParticipantRole</role_name>
                </participant_name></participant_qos>
    </qos_profile>
</qos_library>
</dds>
```

Partitions (XML)




```
<partition>
  <name>
    <element>ABC</element>
  </name>
</partition>
```

How to Refer to QoS Profiles?

```
StringDataWriter dataWriter =
                    (StringDataWriter) participant.create_datawriter_with_profile(
        topic, //Topic
        "Demo1_Library", "Demo1_Profile",
        null,
        StatusKind.DATA_AVAILABLE_STATUS);
```

Anatomy of a DDS Application