```
1  using System.Reflection;
2  using System.Runtime.CompilerServices;
3  using System.Runtime.InteropServices;
4
5  // General Information about an assembly is controlled through the following
6  // Set of attributes. Change these attribute values to modify the information
7  // associated with an assembly.
8  [assembly: AssemblyTitle("RTPS-Impl")]
9  [assembly: AssemblyDescription("")]
10 [assembly: AssemblyConfiguration("")]
11 [assembly: AssemblyCompany("")]
12 [assembly: AssemblyProduct("RTPS-Impl")]
13 [assembly: AssemblyCopyright("Copyright ©  2014")]
14 [assembly: AssemblyTrademark("")]
15 [assembly: AssemblyCulture("")]
16
17 // Setting ComVisible to false makes the types in this assembly not visible
18 // to COM components.  If you need to access a type in this assembly from
19 // COM, Set the ComVisible attribute to true on that type.
20 [assembly: ComVisible(false)]
21
22 // The following GUID is for the ID of the typelib if this project is exposed to COM
23 [assembly: Guid("9102dedd-c311-41bf-ba74-90eb8161e4e8")]
24
25 // Version information for an assembly consists of the following four values:
26 //
27 //      Major Version
28 //      Minor Version
29 //      Build Number
30 //      Revision
31 //
32 // You can specify all the values or you can default the Build and Revision Numbers
33 // by using the '*' as shown below:
34 // [assembly: AssemblyVersion("1.0.*")]
35 [assembly: AssemblyVersion("0.0.*")]
36 [assembly: AssemblyFileVersion("0.0.0.0")]
37
38 [assembly: log4net.Config.XmlConfigurator(ConfigFile = "Log4Net.config", Watch = true)]
```

```
1  using org.omg.dds.core;
2  using System;
3  using System.Collections.Generic;
4  using System.Linq;
5  using System.Text;
6  using System.Threading.Tasks;
7
8  namespace Doopec.DDS.Core
9  {
10     public abstract class EntityQosImpl<UNMOD_SELF, MOD_SELF> : EntityQos<UNMOD_SELF, MOD_SELF>
11         where UNMOD_SELF : EntityQos<UNMOD_SELF, MOD_SELF>
12         where MOD_SELF : UNMOD_SELF
13     {
14         public Bootstrap Boostrap { get; private set; }
15
16         public EntityQosImpl(Bootstrap boostrap)
17         {
18             this.Boostrap = boostrap;
19         }
20
21         public POLICY Get<POLICY>(org.omg.dds.core.policy.QosPolicyId id) where POLICY : org.omg.dds.core↙
       .policy.QosPolicy
22         {
23             throw new NotImplementedException();
24         }
25
26         public org.omg.dds.core.policy.QosPolicy Put(org.omg.dds.core.policy.QosPolicyId key, org.omg.dds↙
       .core.policy.QosPolicy value)
27         {
28             throw new NotImplementedException();
29         }
30
31         public org.omg.dds.core.policy.QosPolicy Remove(object key)
32         {
33             throw new NotImplementedException();
34         }
35
36         public void Clear()
37         {
38             throw new NotImplementedException();
39         }
40
41         public Bootstrap GetBootstrap()
42         {
43             return Boostrap;
44         }
45
46         public void Add(org.omg.dds.core.policy.QosPolicyId key, org.omg.dds.core.policy.QosPolicy value)
47         {
48             throw new NotImplementedException();
49         }
50
51         public bool ContainsKey(org.omg.dds.core.policy.QosPolicyId key)
52         {
53             throw new NotImplementedException();
54         }
55
56         public ICollection<org.omg.dds.core.policy.QosPolicyId> Keys
57         {
58             get { throw new NotImplementedException(); }
59         }
60
61         public bool Remove(org.omg.dds.core.policy.QosPolicyId key)
62         {
63             throw new NotImplementedException();
64         }
65
66         public bool TryGetValue(org.omg.dds.core.policy.QosPolicyId key, out org.omg.dds.core.policy.    ↙
       QosPolicy value)
67         {
68             throw new NotImplementedException();
69         }
70
71         public ICollection<org.omg.dds.core.policy.QosPolicy> Values
```

```
72          {
73              get { throw new NotImplementedException(); }
74          }
75
76          public org.omg.dds.core.policy.QosPolicy this[org.omg.dds.core.policy.QosPolicyId key]
77          {
78              get
79              {
80                  throw new NotImplementedException();
81              }
82              set
83              {
84                  throw new NotImplementedException();
85              }
86          }
87
88          public void Add(KeyValuePair<org.omg.dds.core.policy.QosPolicyId, org.omg.dds.core.policy.
        QosPolicy> item)
89          {
90              throw new NotImplementedException();
91          }
92
93
94          public bool Contains(KeyValuePair<org.omg.dds.core.policy.QosPolicyId, org.omg.dds.core.policy.
        QosPolicy> item)
95          {
96              throw new NotImplementedException();
97          }
98
99          public void CopyTo(KeyValuePair<org.omg.dds.core.policy.QosPolicyId, org.omg.dds.core.policy.
        QosPolicy>[] array, int arrayIndex)
100         {
101             throw new NotImplementedException();
102         }
103
104         public int Count
105         {
106             get { throw new NotImplementedException(); }
107         }
108
109         public bool IsReadOnly
110         {
111             get { throw new NotImplementedException(); }
112         }
113
114         public bool Remove(KeyValuePair<org.omg.dds.core.policy.QosPolicyId, org.omg.dds.core.policy.
        QosPolicy> item)
115         {
116             throw new NotImplementedException();
117         }
118
119         public IEnumerator<KeyValuePair<org.omg.dds.core.policy.QosPolicyId, org.omg.dds.core.policy.
        QosPolicy>> GetEnumerator()
120         {
121             throw new NotImplementedException();
122         }
123
124         System.Collections.IEnumerator System.Collections.IEnumerable.GetEnumerator()
125         {
126             throw new NotImplementedException();
127         }
128
129         public abstract MOD_SELF Modify();
130
131     }
132 }
133
```

```
1  using org.omg.dds.core;
2  using org.omg.dds.core.policy;
3  using org.omg.dds.core.policy.modifiable;
4  using System;
5  using System.Collections.Generic;
6  using System.Linq;
7  using System.Text;
8  using System.Threading.Tasks;
9
10 namespace Doopec.Dds.Core.Policy.modifiable
11 {
12     class ModifiableDataRepresentationQosPolicyImpl : DataRepresentationQosPolicyImpl,
       ModifiableDataRepresentationQosPolicy
13     {
14
15         public ModifiableDataRepresentationQosPolicyImpl(DataRepresentationQosPolicy qos)
16             : base(qos.GetValue(), qos.GetBootstrap())
17         {
18         }
19
20         public ModifiableDataRepresentationQosPolicyImpl(List<short> getValue, Bootstrap boostrap)
21             : base(getValue, boostrap)
22         {
23
24         }
25         public ModifiableDataRepresentationQosPolicy SetValue(List<short> value)
26         {
27             this.ValueQos = value;
28             return this;
29         }
30
31         public ModifiableDataRepresentationQosPolicy SetValue(params short[] value)
32         {
33             throw new NotImplementedException();
34         }
35
36         public ModifiableDataRepresentationQosPolicy CopyFrom(DataRepresentationQosPolicy other)
37         {
38             return new ModifiableDataRepresentationQosPolicyImpl(other);
39         }
40
41         public DataRepresentationQosPolicy FinishModification()
42         {
43             return new DataRepresentationQosPolicyImpl(this.GetValue(), this.GetBootstrap());
44         }
45     }
46 }
47
```

```
1  using org.omg.dds.core;
2  using org.omg.dds.core.policy;
3  using org.omg.dds.core.policy.modifiable;
4  using System;
5  using System.Collections.Generic;
6  using System.Linq;
7  using System.Text;
8  using System.Threading.Tasks;
9
10 namespace Doopec.Dds.Core.Policy.modifiable
11 {
12     class ModifiableDeadlineQosPolicyImpl : DeadlineQosPolicyImpl, ModifiableDeadlineQosPolicy
13     {
14
15
16         public ModifiableDeadlineQosPolicyImpl(DeadlineQosPolicy qos)
17             : base(qos.GetPeriod(),qos.GetBootstrap())
18         {
19         }
20         public ModifiableDeadlineQosPolicyImpl(Duration getPeriod, Bootstrap boostrap)
21             : base(getPeriod, boostrap )
22         {
23
24         }
25
26
27         public ModifiableDeadlineQosPolicy SetPeriod(Duration period)
28         {
29             this.PeriodQos = period;
30             return this;
31         }
32
33         public ModifiableDeadlineQosPolicy SetPeriod(long period, global::DDS.ConversionUtils.TimeUnit
       unit)
34         {
35             throw new NotImplementedException();
36         }
37
38         public ModifiableDeadlineQosPolicy CopyFrom(DeadlineQosPolicy other)
39         {
40             return new ModifiableDeadlineQosPolicyImpl(other.GetPeriod(), other.GetBootstrap());
41         }
42         public DeadlineQosPolicy FinishModification()
43         {
44             return new DeadlineQosPolicyImpl(this.GetPeriod(),this.GetBootstrap());
45         }
46     }
47 }
48
```

```
1  using org.omg.dds.core;
2  using org.omg.dds.core.policy;
3  using org.omg.dds.core.policy.modifiable;
4  using System;
5  using System.Collections.Generic;
6  using System.Linq;
7  using System.Text;
8  using System.Threading.Tasks;
9
10 namespace Doopec.Dds.Core.Policy.modifiable
11 {
12     class ModifiableDestinationOrderQosPolicyImpl : DestinationOrderQosPolicyImpl,
       ModifiableDestinationOrderQosPolicy
13     {
14         public ModifiableDestinationOrderQosPolicyImpl(DestinationOrderQosPolicy qos)
15             : base(qos.GetKind(),qos.GetBootstrap())
16         {
17         }
18
19         public ModifiableDestinationOrderQosPolicyImpl(DestinationOrderQosPolicyKind kind,Bootstrap
       boostrap)
20             : base(kind,boostrap)
21         {
22
23         }
24
25         public ModifiableDestinationOrderQosPolicy SetKind(DestinationOrderQosPolicyKind kind)
26         {
27             this.KindQos = kind;
28             return this;
29         }
30
31         public ModifiableDestinationOrderQosPolicy CopyFrom(DestinationOrderQosPolicy other)
32         {
33             return new ModifiableDestinationOrderQosPolicyImpl(other.GetKind(),other.GetBootstrap ());
34         }
35
36         public DestinationOrderQosPolicy FinishModification()
37         {
38             return new DestinationOrderQosPolicyImpl(this.GetKind(),this.GetBootstrap());
39         }
40     }
41 }
42
```

```
using org.omg.dds.core;
using org.omg.dds.core.policy;
using org.omg.dds.core.policy.modifiable;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Doopec.Dds.Core.Policy.modifiable
{
    class ModifiableDurabilityQosPolicyImpl : DurabilityQosPolicyImpl, ModifiableDurabilityQosPolicy
    {

        public ModifiableDurabilityQosPolicyImpl(DurabilityQosPolicy qos)
            : base(qos.GetKind(),qos.GetBootstrap())
        {
        }

        public ModifiableDurabilityQosPolicyImpl(DurabilityQosPolicyKind kind, Bootstrap boostrap)
            : base(kind,boostrap)
        {

        }
        public ModifiableDurabilityQosPolicy SetKind(DurabilityQosPolicyKind kind)
        {
            this.KindQos = kind;
            return this;
        }

        public ModifiableDurabilityQosPolicy CopyFrom(DurabilityQosPolicy other)
        {
            return new ModifiableDurabilityQosPolicyImpl(other.GetKind(), other.GetBootstrap());
        }

        public DurabilityQosPolicy FinishModification()
        {
            return new DurabilityQosPolicyImpl(this.GetKind(),this.GetBootstrap());
        }
    }
}

```

```
1  using org.omg.dds.core;
2  using org.omg.dds.core.policy;
3  using org.omg.dds.core.policy.modifiable;
4  using System;
5  using System.Collections.Generic;
6  using System.Linq;
7  using System.Text;
8  using System.Threading.Tasks;
9
10 namespace Doopec.Dds.Core.Policy.modifiable
11 {
12     public class ModifiableDurabilityServiceQosPolicyImpl : DurabilityServiceQosPolicyImpl,
       ModifiableDurabilityServiceQosPolicy
13     {
14         public ModifiableDurabilityServiceQosPolicyImpl(DurabilityServiceQosPolicy qos)
15             : base(qos.GetHistoryKind(),qos.GetServiceCleanupDelay(), qos.GetHistoryDepth(),
16             qos.GetMaxSamples(),qos.GetMaxInstances(),qos.GetMaxSamplesPerInstance(),qos.GetBootstrap())
17         {
18         }
19
20         public ModifiableDurabilityServiceQosPolicyImpl(HistoryQosPolicyKind historyKind, Duration
       serviceCleanupDelay, int historyDepth
21             , int maxSamples, int maxInstances,int maxSamplesPerInstance, Bootstrap boostrap)
22             : base(historyKind,serviceCleanupDelay,historyDepth,maxSamples,maxInstances,
       maxSamplesPerInstance,boostrap)
23         {
24
25         }
26
27         public ModifiableDurabilityServiceQosPolicy SetServiceCleanupDelay(Duration serviceCleanupDelay)
28         {
29             this.ServiceCleanupDelay = serviceCleanupDelay;
30             return this;
31         }
32
33         public ModifiableDurabilityServiceQosPolicy SetServiceCleanupDelay(long serviceCleanupDelay,
       global::DDS.ConversionUtils.TimeUnit unit)
34         {
35             throw new NotImplementedException();
36         }
37
38         public ModifiableDurabilityServiceQosPolicy SetHistoryKind(HistoryQosPolicyKind historyKind)
39         {
40             this.HistoryQosPolicyKind = historyKind;
41             return this;
42         }
43
44         public ModifiableDurabilityServiceQosPolicy SetHistoryDepth(int historyDepth)
45         {
46             this.HistoryDepth = historyDepth;
47             return this;
48         }
49
50         public ModifiableDurabilityServiceQosPolicy SetMaxSamples(int maxSamples)
51         {
52             this.MaxInstancesQos=maxSamples ;
53             return this;
54         }
55
56         public ModifiableDurabilityServiceQosPolicy SetMaxInstances(int maxInstances)
57         {
58             this.MaxInstancesQos =maxInstances ;
59             return this;
60         }
61
62         public ModifiableDurabilityServiceQosPolicy SetMaxSamplesPerInstance(int maxSamplesPerInstance)
63         {
64            this.MaxSamplesPerInstanceQos =maxSamplesPerInstance ;
65            return this;
66         }
67
68         public ModifiableDurabilityServiceQosPolicy CopyFrom(DurabilityServiceQosPolicy other)
69         {
70             return new ModifiableDurabilityServiceQosPolicyImpl(other);
```

```
71          }
72
73          public DurabilityServiceQosPolicy FinishModification()
74          {
75              return new DurabilityServiceQosPolicyImpl(this.GetHistoryKind(),this.GetServiceCleanupDelay()
76                  ,this.GetHistoryDepth(),this.GetMaxSamples(),this.GetMaxInstances()
77                  ,this.GetMaxSamplesPerInstance(),this.GetBootstrap());
78          }
79      }
80  }
81
```

```
1  using Doopec.DDS.Core.Policy;
2  using org.omg.dds.core;
3  using org.omg.dds.core.policy;
4  using org.omg.dds.core.policy.modifiable;
5  using System;
6  using System.Collections.Generic;
7  using System.Linq;
8  using System.Text;
9  using System.Threading.Tasks;
10
11 namespace Doopec.Dds.Core.Policy.modifiable
12 {
13     public class ModifiableEntityFactoryQosPolicyImpl : ModifiableEntityFactoryQosPolicy
14     {
15
16         private EntityFactoryQosPolicyImpl innerQos;
17
18         public ModifiableEntityFactoryQosPolicyImpl(EntityFactoryQosPolicy qos)
19         {
20             this.innerQos = qos as EntityFactoryQosPolicyImpl;
21         }
22
23         public ModifiableEntityFactoryQosPolicyImpl(bool autoEnableCreatedEntities, Bootstrap boostrap)
24         {
25             this.innerQos = new EntityFactoryQosPolicyImpl(autoEnableCreatedEntities, boostrap);
26         }
27
28
29         public ModifiableEntityFactoryQosPolicy SetAutoEnableCreatedEntities(bool                          ↙
       autoEnableCreatedEntities)
30         {
31             this.innerQos.AutoenableCreatedEntities = autoEnableCreatedEntities;
32             return this;
33         }
34
35         public ModifiableEntityFactoryQosPolicy CopyFrom(EntityFactoryQosPolicy other)
36         {
37             return new ModifiableEntityFactoryQosPolicyImpl(other.IsAutoEnableCreatedEntities(), other.   ↙
       GetBootstrap());
38         }
39
40         public EntityFactoryQosPolicy FinishModification()
41         {
42             return this.innerQos;
43         }
44
45         public bool IsAutoEnableCreatedEntities()
46         {
47             return innerQos.IsAutoEnableCreatedEntities();
48         }
49
50         public QosPolicyId GetId()
51         {
52             return innerQos.GetId();
53         }
54
55         public Bootstrap GetBootstrap()
56         {
57             return innerQos.GetBootstrap();
58         }
59
60         public ModifiableEntityFactoryQosPolicy Modify()
61         {
62             return this;
63         }
64     }
65 }
66
```

```
1  using org.omg.dds.core.policy;
2  using org.omg.dds.core.policy.modifiable;
3  using System;
4  using System.Collections.Generic;
5  using System.Linq;
6  using System.Text;
7  using System.Threading.Tasks;
8
9  namespace Doopec.Dds.Core.Policy.modifiable
10 {
11     class ModifiableGroupDataQosPolicyImpl : GroupDataQosPolicyImpl, ModifiableGroupDataQosPolicy
12     {
13         public ModifiableGroupDataQosPolicyImpl(GroupDataQosPolicy qos)
14             : base(qos.GetBootstrap())
15         {
16         }
17
18
19         public ModifiableGroupDataQosPolicy SetValue(byte[] value, int offset, int length)
20         {
21             Value = new byte[length];
22             Array.Copy(value, offset, Value, 0, length);
23             return this;
24         }
25
26         public ModifiableGroupDataQosPolicy CopyFrom(GroupDataQosPolicy other)
27         {
28             return new ModifiableGroupDataQosPolicyImpl (other);
29         }
30
31         public GroupDataQosPolicy FinishModification()
32         {
33             return new GroupDataQosPolicyImpl (this.GetBootstrap());
34         }
35     }
36 }
37
```

```
1  using org.omg.dds.core;
2  using org.omg.dds.core.policy;
3  using org.omg.dds.core.policy.modifiable;
4  using System;
5  using System.Collections.Generic;
6  using System.Linq;
7  using System.Text;
8  using System.Threading.Tasks;
9
10 namespace Doopec.Dds.Core.Policy.modifiable
11 {
12     class ModifiableHistoryQosPolicyImpl : HistoryQosPolicyImpl, ModifiableHistoryQosPolicy
13     {
14         public ModifiableHistoryQosPolicyImpl(HistoryQosPolicy qos)
15             : base(qos.GetKind(), qos.GetDepth(),qos.GetBootstrap())
16         {
17         }
18
19         public ModifiableHistoryQosPolicyImpl(HistoryQosPolicyKind kind, int getDepth,Bootstrap boostrap)
20             : base(kind, getDepth,boostrap)
21         {
22
23         }
24         public ModifiableHistoryQosPolicy SetKind(HistoryQosPolicyKind kind)
25         {
26             this.KindQos=kind;
27             return this;
28         }
29
30         public ModifiableHistoryQosPolicy SetDepth(int depth)
31         {
32             this.DepthQos=depth;
33             return this;
34         }
35
36         public ModifiableHistoryQosPolicy CopyFrom(HistoryQosPolicy other)
37         {
38             return new ModifiableHistoryQosPolicyImpl(other.GetKind(),other.GetDepth()
39                 ,other.GetBootstrap());
40         }
41
42         public HistoryQosPolicy FinishModification()
43         {
44             return new HistoryQosPolicyImpl(this.GetKind(),this.GetDepth(),this.GetBootstrap());
45         }
46     }
47 }
48
```

```
1  using org.omg.dds.core;
2  using org.omg.dds.core.policy;
3  using org.omg.dds.core.policy.modifiable;
4  using System;
5  using System.Collections.Generic;
6  using System.Linq;
7  using System.Text;
8  using System.Threading.Tasks;
9
10 namespace Doopec.Dds.Core.Policy.modifiable
11 {
12     class ModifiableLatencyBudgetQosPolicyImpl: LatencyBudgetQosPolicyImpl,
       ModifiableLatencyBudgetQosPolicy
13     {
14         public ModifiableLatencyBudgetQosPolicyImpl(LatencyBudgetQosPolicy qos)
15             : base(qos.GetDuration(),qos.GetBootstrap())
16         {
17         }
18
19         public ModifiableLatencyBudgetQosPolicyImpl(Duration duration,Bootstrap boostrap)
20             : base(duration, boostrap)
21         {
22
23         }
24
25         public ModifiableLatencyBudgetQosPolicy SetDuration(Duration duration)
26         {
27             this.GetDurationQos=duration;
28             return this;
29         }
30
31         public ModifiableLatencyBudgetQosPolicy SetDuration(long duration, global::DDS.ConversionUtils.
       TimeUnit unit)
32         {
33             throw new NotImplementedException();
34         }
35
36         public ModifiableLatencyBudgetQosPolicy CopyFrom(LatencyBudgetQosPolicy other)
37         {
38             return new ModifiableLatencyBudgetQosPolicyImpl (other);
39         }
40
41         public LatencyBudgetQosPolicy FinishModification()
42         {
43             return new LatencyBudgetQosPolicyImpl(this.GetDuration(),this.GetBootstrap());
44         }
45     }
46 }
47
```

```
1  using org.omg.dds.core;
2  using org.omg.dds.core.policy;
3  using org.omg.dds.core.policy.modifiable;
4  using System;
5  using System.Collections.Generic;
6  using System.Linq;
7  using System.Text;
8  using System.Threading.Tasks;
9
10 namespace Doopec.Dds.Core.Policy.modifiable
11 {
12     class ModifiableLifespanQosPolicyImpl : LifespanQosPolicyImpl, ModifiableLifespanQosPolicy
13     {
14         public ModifiableLifespanQosPolicyImpl(LifespanQosPolicy qos)
15             : base(qos.GetDuration(),qos.GetBootstrap())
16         {
17         }
18
19         public ModifiableLifespanQosPolicyImpl(Duration duration,Bootstrap boostrap)
20             : base(duration,boostrap )
21         {
22
23         }
24
25
26         public ModifiableLifespanQosPolicy SetDuration(Duration duration)
27         {
28             this.GetDurationQos = duration;
29             return this;
30         }
31
32         public ModifiableLifespanQosPolicy SetDuration(long duration, global::DDS.ConversionUtils.TimeUnit↙
       unit)
33         {
34             throw new NotImplementedException();
35         }
36
37         public ModifiableLifespanQosPolicy CopyFrom(LifespanQosPolicy other)
38         {
39             return new ModifiableLifespanQosPolicyImpl(other);
40         }
41
42         public LifespanQosPolicy FinishModification()
43         {
44             return new LifespanQosPolicyImpl (this.GetDuration(),this.GetBootstrap());
45         }
46     }
47 }
48
```

```
1  using org.omg.dds.core;
2  using org.omg.dds.core.policy;
3  using org.omg.dds.core.policy.modifiable;
4  using System;
5  using System.Collections.Generic;
6  using System.Linq;
7  using System.Text;
8  using System.Threading.Tasks;
9
10 namespace Doopec.Dds.Core.Policy.modifiable
11 {
12     class ModifiableLivelinessQosPolicyImpl : LivelinessQosPolicyImpl, ModifiableLivelinessQosPolicy
13     {
14         public ModifiableLivelinessQosPolicyImpl(LivelinessQosPolicy qos)
15             : base(qos.GetKind(),qos.GetLeaseDuration(),qos.GetBootstrap())
16         {
17         }
18
19         public ModifiableLivelinessQosPolicyImpl(LivelinessQosPolicyKind kind, Duration leaseDuration,
       Bootstrap boostrap)
20             : base(kind,leaseDuration,boostrap)
21         {
22
23         }
24
25
26         public ModifiableLivelinessQosPolicy SetKind(LivelinessQosPolicyKind kind)
27         {
28             this.KindQos =kind ;
29             return this;
30         }
31
32         public ModifiableLivelinessQosPolicy SetLeaseDuration(Duration leaseDuration)
33         {
34             this.LeaseDurationQos =leaseDuration ;
35             return this;
36         }
37
38         public ModifiableLivelinessQosPolicy SetLeaseDuration(long leaseDuration, global::DDS.
       ConversionUtils.TimeUnit unit)
39         {
40             throw new NotImplementedException();
41         }
42
43         public ModifiableLivelinessQosPolicy CopyFrom(LivelinessQosPolicy other)
44         {
45             return new ModifiableLivelinessQosPolicyImpl (other.GetKind(),other.GetLeaseDuration()
46                 ,other.GetBootstrap());
47         }
48
49         public LivelinessQosPolicy FinishModification()
50         {
51             return new LivelinessQosPolicyImpl (this.GetKind(),this.GetLeaseDuration(),this.GetBootstrap
       ());
52         }
53     }
54 }
55
```

```
1  using org.omg.dds.core;
2  using org.omg.dds.core.policy;
3  using org.omg.dds.core.policy.modifiable;
4  using System;
5  using System.Collections.Generic;
6  using System.Linq;
7  using System.Text;
8  using System.Threading.Tasks;
9
10 namespace Doopec.Dds.Core.Policy.modifiable
11 {
12     class ModifiableOwnershipQosPolicyImpl : OwnershipQosPolicyImpl, ModifiableOwnershipQosPolicy
13     {
14         public ModifiableOwnershipQosPolicyImpl(OwnershipQosPolicy qos)
15             : base(qos.GetKind(),qos.GetBootstrap())
16         {
17         }
18
19         public ModifiableOwnershipQosPolicyImpl(OwnershipQosPolicyKind kind,Bootstrap boostrap)
20             : base(kind,boostrap)
21         {
22
23         }
24         public ModifiableOwnershipQosPolicy SetKind(OwnershipQosPolicyKind kind)
25         {
26             this.KindQos=kind ;
27             return this;
28         }
29
30
31
32
33         public ModifiableOwnershipQosPolicy CopyFrom(OwnershipQosPolicy other)
34         {
35             return new ModifiableOwnershipQosPolicyImpl(other);
36         }
37
38         public OwnershipQosPolicy FinishModification()
39         {
40             return new OwnershipQosPolicyImpl(this.GetKind(),this.GetBootstrap());
41         }
42     }
43 }
44
```

```
using org.omg.dds.core;
using org.omg.dds.core.policy;
using org.omg.dds.core.policy.modifiable;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Doopec.Dds.Core.Policy.modifiable
{
    class ModifiableOwnershipStrengthQosPolicyImpl : OwnershipStrengthQosPolicyImpl,
    ModifiableOwnershipStrengthQosPolicy
    {
        public ModifiableOwnershipStrengthQosPolicyImpl(OwnershipStrengthQosPolicy qos)
            : base(qos.GetValue(),qos.GetBootstrap())
        {
        }

        public ModifiableOwnershipStrengthQosPolicyImpl(int strength, Bootstrap boostrap)
            : base(strength,boostrap)
        {
        }

        public ModifiableOwnershipStrengthQosPolicy SetValue(int value)
        {
            this.StrengthQos = value;
            return this;
        }

        public ModifiableOwnershipStrengthQosPolicy CopyFrom(OwnershipStrengthQosPolicy other)
        {
            return new ModifiableOwnershipStrengthQosPolicyImpl(other);
        }

        public OwnershipStrengthQosPolicy FinishModification()
        {
            return new OwnershipStrengthQosPolicyImpl(this.GetValue(),this.GetBootstrap());
        }
    }
}

```

```
1  using org.omg.dds.core;
2  using org.omg.dds.core.policy;
3  using org.omg.dds.core.policy.modifiable;
4  using System;
5  using System.Collections.Generic;
6  using System.Linq;
7  using System.Text;
8  using System.Threading.Tasks;
9
10 namespace Doopec.Dds.Core.Policy.modifiable
11 {
12     class ModifiablePartitionQosPolicyImpl : PartitionQosPolicyImpl, ModifiablePartitionQosPolicy
13     {
14           public ModifiablePartitionQosPolicyImpl(PartitionQosPolicy qos)
15             : base(qos.GetName(), qos.GetBootstrap())
16         {
17         }
18
19           public ModifiablePartitionQosPolicyImpl(ICollection<string> name, Bootstrap boostrap)
20             : base(name, boostrap)
21         {
22
23         }
24
25
26         public ModifiablePartitionQosPolicy SetName(ICollection<string> name)
27         {
28             this.NameQos=name;
29             return this;
30         }
31
32         public ModifiablePartitionQosPolicy CopyFrom(PartitionQosPolicy other)
33         {
34             return new ModifiablePartitionQosPolicyImpl(other.GetName(),other.GetBootstrap());
35         }
36
37         public PartitionQosPolicy FinishModification()
38         {
39             return new PartitionQosPolicyImpl(this.GetName(),this.GetBootstrap());
40         }
41     }
42 }
43
```

```
1  using org.omg.dds.core;
2  using org.omg.dds.core.policy;
3  using org.omg.dds.core.policy.modifiable;
4  using System;
5  using System.Collections.Generic;
6  using System.Linq;
7  using System.Text;
8  using System.Threading.Tasks;
9
10 namespace Doopec.Dds.Core.Policy.modifiable
11 {
12     class ModifiablePresentationQosPolicyImpl: PresentationQosPolicyImpl, ModifiablePresentationQosPolicy
13     {
14           public ModifiablePresentationQosPolicyImpl(PresentationQosPolicy qos)
15             : base(qos.GetAccessScope(),qos.IsCoherentAccess(),qos.IsOrderedAccess(), qos.GetBootstrap())
16         {
17         }
18
19           public ModifiablePresentationQosPolicyImpl(AccessScopeKind accessScope,bool coherentAccess,bool ↙
       orderedAccess, Bootstrap boostrap)
20             : base(accessScope,coherentAccess,orderedAccess , boostrap)
21         {
22
23         }
24
25
26           public ModifiablePresentationQosPolicy SetAccessScope(AccessScopeKind accessScope)
27           {
28               this.AccessScopeKindQos=accessScope ;
29               return this;
30           }
31
32           public ModifiablePresentationQosPolicy SetCoherentAccess(bool coherentAccess)
33           {
34               this.IsCoherentAccessQos=coherentAccess ;
35               return this;
36           }
37
38           public ModifiablePresentationQosPolicy SetOrderedAccess(bool orderedAccess)
39           {
40               this.IsOrderedAccessQos =orderedAccess ;
41               return this;
42           }
43
44           public ModifiablePresentationQosPolicy CopyFrom(PresentationQosPolicy other)
45           {
46               return new ModifiablePresentationQosPolicyImpl (other);
47           }
48
49           public PresentationQosPolicy FinishModification()
50           {
51               return new PresentationQosPolicyImpl (this.GetAccessScope(),this.IsCoherentAccess(),this.   ↙
       IsOrderedAccess(),this.GetBootstrap());
52           }
53     }
54 }
55
```

```
1  using org.omg.dds.core;
2  using org.omg.dds.core.policy;
3  using org.omg.dds.core.policy.modifiable;
4  using System;
5  using System.Collections.Generic;
6  using System.Linq;
7  using System.Text;
8  using System.Threading.Tasks;
9
10 namespace Doopec.Dds.Core.Policy.modifiable
11 {
12     class ModifiableQosPolicyCountImpl : QosPolicyCountImpl, QosPolicyCount
13     {
14         public ModifiableQosPolicyCountImpl(QosPolicyCount qos)
15             : base(qos.GetBootstrap())
16         {
17         }
18
19         public ModifiableQosPolicyCountImpl(Bootstrap boostrap)
20             : base(boostrap)
21         {
22
23         }
24     }
25 }
26
```

```
1  using org.omg.dds.core;
2  using org.omg.dds.core.policy;
3  using org.omg.dds.core.policy.modifiable;
4  using System;
5  using System.Collections.Generic;
6  using System.Linq;
7  using System.Text;
8  using System.Threading.Tasks;
9
10 namespace Doopec.Dds.Core.Policy.modifiable
11 {
12     class ModifiableReaderDataLifecycleQosPolicyImpl : ReaderDataLifecycleQosPolicyImpl,
       ModifiableReaderDataLifecycleQosPolicy
13     {
14         public ModifiableReaderDataLifecycleQosPolicyImpl(ReaderDataLifecycleQosPolicy qos)
15             : base(qos.GetAutoPurgeNoWriterSamplesDelay(),qos.GetAutoPurgeDisposedSamplesDelay(), qos.
       GetBootstrap())
16         {
17         }
18
19         public ModifiableReaderDataLifecycleQosPolicyImpl(Duration autoPurgeNoWriterSamplesDelay, Duration
        autoPurgeDisposedSamplesDelay, Bootstrap boostrap)
20             : base(autoPurgeNoWriterSamplesDelay,autoPurgeDisposedSamplesDelay, boostrap)
21         {
22
23         }
24
25
26         public ModifiableReaderDataLifecycleQosPolicy SetAutoPurgeNoWriterSamplesDelay(Duration
       autoPurgeNoWriterSamplesDelay)
27         {
28             this.AutoPurgeNoWriterSamplesDelay=autoPurgeNoWriterSamplesDelay;
29             return this;
30         }
31
32         public ModifiableReaderDataLifecycleQosPolicy SetAutoPurgeNoWriterSamplesDelay(long
       autoPurgeNoWriterSamplesDelay, global::DDS.ConversionUtils.TimeUnit unit)
33         {
34             throw new NotImplementedException();
35         }
36
37         public ModifiableReaderDataLifecycleQosPolicy SetAutoPurgeDisposedSamplesDelay(Duration
       autoPurgeDisposedSamplesDelay)
38         {
39             this.AutoPurgeDisposedSamplesDelay=autoPurgeDisposedSamplesDelay;
40             return this;
41         }
42
43         public ModifiableReaderDataLifecycleQosPolicy SetAutoPurgeDisposedSamplesDelay(long
       autoPurgeDisposedSamplesDelay, global::DDS.ConversionUtils.TimeUnit unit)
44         {
45             throw new NotImplementedException();
46         }
47
48         public ModifiableReaderDataLifecycleQosPolicy CopyFrom(ReaderDataLifecycleQosPolicy other)
49         {
50             return new ModifiableReaderDataLifecycleQosPolicyImpl (other);
51         }
52
53         public ReaderDataLifecycleQosPolicy FinishModification()
54         {
55             return new ReaderDataLifecycleQosPolicyImpl (this.GetAutoPurgeNoWriterSamplesDelay(),this.
       GetAutoPurgeDisposedSamplesDelay(),this.GetBootstrap());
56         }
57     }
58 }
59
```

```
1  using org.omg.dds.core;
2  using org.omg.dds.core.policy;
3  using org.omg.dds.core.policy.modifiable;
4  using System;
5  using System.Collections.Generic;
6  using System.Linq;
7  using System.Text;
8  using System.Threading.Tasks;
9
10 namespace Doopec.Dds.Core.Policy.modifiable
11 {
12     class ModifiableReliabilityQosPolicyImpl : ReliabilityQosPolicyImpl, ModifiableReliabilityQosPolicy
13     {
14         public ModifiableReliabilityQosPolicyImpl(ReliabilityQosPolicy qos)
15             : base(qos.GetKind(),qos.GetMaxBlockingTime(),qos.GetBootstrap())
16         {
17         }
18
19         public ModifiableReliabilityQosPolicyImpl(ReliabilityQosPolicyKind kind, Duration maxBlockingTime,↙
       Bootstrap boostrap)
20             : base(kind,maxBlockingTime, boostrap)
21         {
22
23         }
24
25         public ModifiableReliabilityQosPolicy SetKind(ReliabilityQosPolicyKind kind)
26         {
27             this.KindQos=kind ;
28             return this;
29         }
30
31         public ModifiableReliabilityQosPolicy SetMaxBlockingTime(Duration maxBlockingTime)
32         {
33             this.MaxBlockingTimeQos =maxBlockingTime;
34             return this;
35         }
36
37         public ModifiableReliabilityQosPolicy SetMaxBlockingTime(long maxBlockingTime, global::DDS.       ↙
       ConversionUtils.TimeUnit unit)
38         {
39             throw new NotImplementedException();
40         }
41
42         public ModifiableReliabilityQosPolicy CopyFrom(ReliabilityQosPolicy other)
43         {
44             return new ModifiableReliabilityQosPolicyImpl(other) ;
45         }
46
47         public ReliabilityQosPolicy FinishModification()
48         {
49             return new ReliabilityQosPolicyImpl(this.GetKind(), this.GetMaxBlockingTime(), this.          ↙
       GetBootstrap());
50         }
51     }
52 }
53
```

```
1  using org.omg.dds.core;
2  using org.omg.dds.core.policy;
3  using org.omg.dds.core.policy.modifiable;
4  using System;
5  using System.Collections.Generic;
6  using System.Linq;
7  using System.Text;
8  using System.Threading.Tasks;
9
10 namespace Doopec.Dds.Core.Policy.modifiable
11 {
12     class ModifiableResourceLimitsQosPolicyImpl : ResourceLimitsQosPolicyImpl,
       ModifiableResourceLimitsQosPolicy
13     {
14         public ModifiableResourceLimitsQosPolicyImpl(ResourceLimitsQosPolicy qos)
15             : base(qos.GetMaxSamples(),qos.GetMaxInstances(),qos.GetMaxSamplesPerInstance(),qos.
       GetBootstrap())
16         {
17         }
18
19         public ModifiableResourceLimitsQosPolicyImpl(int maxSamples, int maxInstances, int
       maxSamplesPerInstance, Bootstrap boostrap)
20             : base(maxSamples,maxInstances,maxSamplesPerInstance , boostrap)
21         {
22
23         }
24
25         public ModifiableResourceLimitsQosPolicy SetMaxSamples(int maxSamples)
26         {
27             this.MaxSamplesQos =maxSamples ;
28             return this;
29         }
30
31         public ModifiableResourceLimitsQosPolicy SetMaxInstances(int maxInstances)
32         {
33             this.MaxInstancesQos =maxInstances ;
34             return this;
35         }
36
37         public ModifiableResourceLimitsQosPolicy SetMaxSamplesPerInstance(int maxSamplesPerInstance)
38         {
39             this.MaxSamplesPerInstanceQos =maxSamplesPerInstance ;
40             return this;
41         }
42
43         public ModifiableResourceLimitsQosPolicy CopyFrom(ResourceLimitsQosPolicy other)
44         {
45             return new ModifiableResourceLimitsQosPolicyImpl(other);
46         }
47
48         public ResourceLimitsQosPolicy FinishModification()
49         {
50             return new ResourceLimitsQosPolicyImpl (this.GetMaxSamples (),this.GetMaxInstances (),
51                 this.GetMaxSamplesPerInstance(),this.GetBootstrap());
52         }
53     }
54 }
55
```

```
1  using org.omg.dds.core;
2  using org.omg.dds.core.policy;
3  using org.omg.dds.core.policy.modifiable;
4  using System;
5  using System.Collections.Generic;
6  using System.Linq;
7  using System.Text;
8  using System.Threading.Tasks;
9
10 namespace Doopec.Dds.Core.Policy.modifiable
11 {
12     class ModifiableTimeBasedFilterQosPolicyImpl : TimeBasedFilterQosPolicyImpl,
       ModifiableTimeBasedFilterQosPolicy
13     {
14         public ModifiableTimeBasedFilterQosPolicyImpl(TimeBasedFilterQosPolicy qos)
15             : base(qos.GetMinimumSeparation(),qos.GetBootstrap())
16         {
17         }
18
19         public ModifiableTimeBasedFilterQosPolicyImpl(Duration minimumSeparation, Bootstrap boostrap)
20             : base(minimumSeparation , boostrap)
21         {
22
23         }
24
25         public ModifiableTimeBasedFilterQosPolicy SetMinimumSeparation(Duration minimumSeparation)
26         {
27             this.MinimumSeparationQos =minimumSeparation ;
28             return this;
29         }
30
31         public ModifiableTimeBasedFilterQosPolicy SetMinimumSeparation(long minimumSeparation, global::DDS
       .ConversionUtils.TimeUnit unit)
32         {
33             throw new NotImplementedException();
34         }
35
36         public ModifiableTimeBasedFilterQosPolicy CopyFrom(TimeBasedFilterQosPolicy other)
37         {
38             return new ModifiableTimeBasedFilterQosPolicyImpl (other);
39         }
40
41         public TimeBasedFilterQosPolicy FinishModification()
42         {
43             return new TimeBasedFilterQosPolicyImpl(this.GetMinimumSeparation(), this.GetBootstrap());
44         }
45     }
46 }
47
```

```
1  using org.omg.dds.core.policy;
2  using org.omg.dds.core.policy.modifiable;
3  using System;
4  using System.Collections.Generic;
5  using System.Linq;
6  using System.Text;
7  using System.Threading.Tasks;
8
9  namespace Doopec.Dds.Core.Policy.modifiable
10 {
11     class ModifiableTopicDataQosPolicyImpl : TopicDataQosPolicyImpl, ModifiableTopicDataQosPolicy
12     {
13         public ModifiableTopicDataQosPolicyImpl(TopicDataQosPolicy qos)
14             : base(qos.GetBootstrap())
15         {
16         }
17
18
19         public ModifiableTopicDataQosPolicy SetValue(byte[] value, int offset, int length)
20         {
21             ValueQos = new byte[length];
22             Array.Copy(value, offset, ValueQos, 0, length);
23             return this;
24         }
25
26         public ModifiableTopicDataQosPolicy CopyFrom(TopicDataQosPolicy other)
27         {
28             return new ModifiableTopicDataQosPolicyImpl (other);
29         }
30
31         public TopicDataQosPolicy FinishModification()
32         {
33             return new TopicDataQosPolicyImpl(this.GetBootstrap());
34         }
35     }
36 }
37
```

```
1  using org.omg.dds.core;
2  using org.omg.dds.core.policy;
3  using org.omg.dds.core.policy.modifiable;
4  using System;
5  using System.Collections.Generic;
6  using System.Linq;
7  using System.Text;
8  using System.Threading.Tasks;
9
10 namespace Doopec.Dds.Core.Policy.modifiable
11 {
12     class ModifiableTransportPriorityQosPolicyImpl:TransportPriorityQosPolicyImpl,
       ModifiableTransportPriorityQosPolicy
13     {
14         public ModifiableTransportPriorityQosPolicyImpl(TransportPriorityQosPolicy qos)
15             : base(qos.GetValue(),qos.GetBootstrap())
16         {
17         }
18
19         public ModifiableTransportPriorityQosPolicyImpl(int value, Bootstrap boostrap)
20             : base(value,boostrap)
21         {
22
23         }
24
25         public ModifiableTransportPriorityQosPolicy SetValue(int value)
26         {
27             this.ValueQos =value;
28             return this;
29         }
30
31         public ModifiableTransportPriorityQosPolicy CopyFrom(TransportPriorityQosPolicy other)
32         {
33             return new ModifiableTransportPriorityQosPolicyImpl(other);
34         }
35
36         public TransportPriorityQosPolicy FinishModification()
37         {
38             return new TransportPriorityQosPolicyImpl(this.GetValue(),this.GetBootstrap());
39         }
40     }
41 }
42
```

```
using org.omg.dds.core;
using org.omg.dds.core.policy;
using org.omg.dds.core.policy.modifiable;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Doopec.Dds.Core.Policy.modifiable
{
    class ModifiableTypeConsistencyEnforcementQosPolicyImpl : TypeConsistencyEnforcementQosPolicyImpl, ModifiableTypeConsistencyEnforcementQosPolicy
    {
        public ModifiableTypeConsistencyEnforcementQosPolicyImpl(TypeConsistencyEnforcementQosPolicy qos)
            : base(qos.GetKind(),qos.GetBootstrap())
        {
        }

        public ModifiableTypeConsistencyEnforcementQosPolicyImpl(TypeConsistencyEnforcementQosPolicyKind kind, Bootstrap boostrap)
            : base(kind,boostrap)
        {

        }

        public ModifiableTypeConsistencyEnforcementQosPolicy SetKind (TypeConsistencyEnforcementQosPolicyKind kind)
        {
            this.KindQos=kind;
            return this;
        }

        public ModifiableTypeConsistencyEnforcementQosPolicy CopyFrom(TypeConsistencyEnforcementQosPolicy other)
        {
            return new ModifiableTypeConsistencyEnforcementQosPolicyImpl (other);
        }

        public TypeConsistencyEnforcementQosPolicy FinishModification()
        {
            return new TypeConsistencyEnforcementQosPolicyImpl(this.GetKind(),this.GetBootstrap());
        }
    }
}

```

```
1  using Doopec.DDS.Core.Policy;
2  using org.omg.dds.core.policy;
3  using org.omg.dds.core.policy.modifiable;
4  using System;
5  using System.Collections.Generic;
6  using System.Linq;
7  using System.Text;
8  using System.Threading.Tasks;
9
10 namespace Doopec.Dds.Core.Policy.modifiable
11 {
12     class ModifiableUserDataQosPolicyImpl: UserDataQosPolicyImpl, ModifiableUserDataQosPolicy
13     {
14         public ModifiableUserDataQosPolicyImpl(UserDataQosPolicy qos)
15             : base(qos.GetBootstrap())
16         {
17         }
18
19         public ModifiableUserDataQosPolicy SetValue(byte[] value, int offset, int length)
20         {
21             ValueQos = new byte[length];
22             Array.Copy(value, offset, ValueQos, 0, length);
23             return this;
24         }
25
26         public ModifiableUserDataQosPolicy CopyFrom(UserDataQosPolicy other)
27         {
28             return new ModifiableUserDataQosPolicyImpl(other);
29         }
30
31         public UserDataQosPolicy FinishModification()
32         {
33             return new UserDataQosPolicyImpl(this.GetBootstrap());
34         }
35     }
36 }
37
```

```
1  using org.omg.dds.core;
2  using org.omg.dds.core.policy;
3  using org.omg.dds.core.policy.modifiable;
4  using System;
5  using System.Collections.Generic;
6  using System.Linq;
7  using System.Text;
8  using System.Threading.Tasks;
9
10 namespace Doopec.Dds.Core.Policy.modifiable
11 {
12     class ModifiableWriterDataLifecycleQosPolicyImpl:WriterDataLifecycleQosPolicyImpl,
       ModifiableWriterDataLifecycleQosPolicy
13     {
14         public ModifiableWriterDataLifecycleQosPolicyImpl(WriterDataLifecycleQosPolicy qos)
15             : base(qos.IsAutDisposeUnregisteredInstances(),qos.GetBootstrap())
16         {
17         }
18
19         public ModifiableWriterDataLifecycleQosPolicyImpl(bool autDisposeUnregisteredInstances, Bootstrap
       boostrap)
20             : base(autDisposeUnregisteredInstances,boostrap)
21         {
22
23         }
24
25         public ModifiableWriterDataLifecycleQosPolicy SetAutDisposeUnregisteredInstances(bool
       autDisposeUnregisteredInstances)
26         {
27             this.IsAutDisposeUnregisteredInstancesQos=autDisposeUnregisteredInstances ;
28             return this;
29         }
30
31         public ModifiableWriterDataLifecycleQosPolicy CopyFrom(WriterDataLifecycleQosPolicy other)
32         {
33             return new ModifiableWriterDataLifecycleQosPolicyImpl(other);
34         }
35
36         public WriterDataLifecycleQosPolicy FinishModification()
37         {
38             return new WriterDataLifecycleQosPolicyImpl(this.IsAutDisposeUnregisteredInstances(),this.
       GetBootstrap());
39         }
40     }
41 }
42
```

```
1  using Doopec.Dds.Core.Policy.modifiable;
2  using Doopec.DDS.Core.Policy;
3  using org.omg.dds.core;
4  using org.omg.dds.core.policy;
5  using org.omg.dds.core.policy.modifiable;
6  using System;
7  using System.Collections.Generic;
8  using System.Linq;
9  using System.Text;
10 using System.Threading.Tasks;
11
12 namespace Doopec.Dds.Core.Policy
13 {
14     public class DataRepresentationQosPolicyImpl : QosPolicyImpl,DataRepresentationQosPolicy
15     {
16
17         public List<short> ValueQos { get; protected internal set; }
18
19         public DataRepresentationQosPolicyImpl(Bootstrap boostrap)
20             : base(boostrap)
21         {
22         }
23
24         public DataRepresentationQosPolicyImpl(List<short> getValue, Bootstrap boostrap)
25             : base(boostrap)
26         {
27             this.ValueQos = getValue;
28         }
29
30         public List<short> GetValue()
31         {
32             return ValueQos;
33         }
34
35         public QosPolicyId GetId()
36         {
37             throw new NotImplementedException();
38         }
39
40
41         public ModifiableDataRepresentationQosPolicy Modify()
42         {
43             return new ModifiableDataRepresentationQosPolicyImpl(this);
44         }
45     }
46 }
47
```

```
using org.omg.dds.core;
using org.omg.dds.core.policy;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using org.omg.dds.core.policy.modifiable;
using Doopec.Dds.Core.Policy.modifiable;
using Doopec.DDS.Core.Policy;




namespace Doopec.Dds.Core.Policy
{
    public class DeadlineQosPolicyImpl : QosPolicyImpl, DeadlineQosPolicy
    {


        public Duration PeriodQos { get; protected internal set; }

         public DeadlineQosPolicyImpl(Bootstrap boostrap)
            : base(boostrap)
        {
        }

         public DeadlineQosPolicyImpl(Duration getPeriod, Bootstrap boostrap)
            : base(boostrap)
        {
            this.PeriodQos = getPeriod;
        }



        public Duration GetPeriod()
        {
            return PeriodQos;
        }



        public ModifiableDeadlineQosPolicy Modify()
        {

            return new ModifiableDeadlineQosPolicyImpl(this);
        }
    }
}

```

```
1  using org.omg.dds.core.policy;
2  using org.omg.dds.core.policy.modifiable;
3  using org.omg.dds.core;
4  using System;
5  using System.Collections.Generic;
6  using System.Linq;
7  using System.Text;
8  using System.Threading.Tasks;
9  using Doopec.Dds.Core.Policy.modifiable;
10 using Doopec.DDS.Core.Policy;
11
12 namespace Doopec.Dds.Core.Policy
13 {
14     public class DestinationOrderQosPolicyImpl: QosPolicyImpl, DestinationOrderQosPolicy
15     {
16
17         public DestinationOrderQosPolicyKind KindQos { get; protected internal set; }
18
19
20         public DestinationOrderQosPolicyImpl(Bootstrap boostrap)
21             : base(boostrap)
22         {
23         }
24
25         public DestinationOrderQosPolicyImpl(DestinationOrderQosPolicyKind kind, Bootstrap boostrap)
26             : base(boostrap)
27         {
28             this.KindQos = kind;
29         }
30         public DestinationOrderQosPolicyKind GetKind()
31         {
32             return KindQos;
33         }
34
35
36         public ModifiableDestinationOrderQosPolicy Modify()
37         {
38             return new ModifiableDestinationOrderQosPolicyImpl(this);
39         }
40     }
41 }
42
```

```
using Doopec.Dds.Core.Policy.modifiable;
using Doopec.DDS.Core.Policy;
using org.omg.dds.core;
using org.omg.dds.core.policy;
using org.omg.dds.core.policy.modifiable;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Doopec.Dds.Core.Policy
{
    public class DurabilityQosPolicyImpl : QosPolicyImpl, DurabilityQosPolicy
    {

        public DurabilityQosPolicyKind KindQos {get; protected internal set; }
         public DurabilityQosPolicyImpl(Bootstrap boostrap)
            : base(boostrap)
        {
        }
        public DurabilityQosPolicyImpl(DurabilityQosPolicyKind kind, Bootstrap boostrap)
            : base(boostrap)
        {
            this.KindQos = kind;
        }

        public DurabilityQosPolicyKind GetKind()
        {
            return KindQos;
        }

        public ModifiableDurabilityQosPolicy Modify()
        {
            return new ModifiableDurabilityQosPolicyImpl(this);
        }
    }
}

```

```
1  using Doopec.Dds.Core.Policy.modifiable;
2  using Doopec.DDS.Core.Policy;
3  using org.omg.dds.core;
4  using org.omg.dds.core.policy;
5  using org.omg.dds.core.policy.modifiable;
6  using System;
7  using System.Collections.Generic;
8  using System.Linq;
9  using System.Text;
10 using System.Threading.Tasks;
11
12 namespace Doopec.Dds.Core.Policy
13 {
14     public class DurabilityServiceQosPolicyImpl : QosPolicyImpl, DurabilityServiceQosPolicy
15     {
16         public HistoryQosPolicyKind HistoryQosPolicyKind { get; protected internal set; }
17         public Duration ServiceCleanupDelay { get; protected internal set; }
18
19         public int HistoryDepth { get; protected internal set; }
20
21         public int MaxSamplesQos { get; protected internal set; }
22         public int MaxInstancesQos { get; protected internal set; }
23         public int MaxSamplesPerInstanceQos { get; protected internal set; }
24
25
26         public DurabilityServiceQosPolicyImpl(Bootstrap boostrap)
27             : base(boostrap)
28         {
29         }
30
31         public DurabilityServiceQosPolicyImpl(HistoryQosPolicyKind kind, Duration serviceCleanupDelay
32             , int historyDepth, int getMaxSamplesQos, int getMaxInstancesQos, int                    ↙
       getMaxSamplesPerInstanceQos, Bootstrap boostrap)
33             : base(boostrap)
34         {
35             this.HistoryQosPolicyKind = kind;
36             this.ServiceCleanupDelay = serviceCleanupDelay;
37             this.HistoryDepth = historyDepth;
38             this.MaxSamplesQos = getMaxSamplesQos;
39             this.MaxInstancesQos = getMaxInstancesQos;
40             this.MaxSamplesPerInstanceQos = getMaxSamplesPerInstanceQos;
41         }
42
43
44         public Duration GetServiceCleanupDelay()
45         {
46             return ServiceCleanupDelay;
47         }
48
49         public HistoryQosPolicyKind GetHistoryKind()
50         {
51             return HistoryQosPolicyKind;
52         }
53
54         public int GetHistoryDepth()
55         {
56             return HistoryDepth;
57         }
58
59         public int GetMaxSamples()
60         {
61             return MaxSamplesQos;
62         }
63
64         public int GetMaxInstances()
65         {
66             return MaxInstancesQos;
67         }
68
69         public int GetMaxSamplesPerInstance()
70         {
71             return MaxSamplesPerInstanceQos;
72         }
73
```

```
74          public ModifiableDurabilityServiceQosPolicy Modify()
75          {
76              return new ModifiableDurabilityServiceQosPolicyImpl(this);
77          }
78      }
79  }
80
```

```
1  using Doopec.Dds.Core.Policy.modifiable;
2  using org.omg.dds.core;
3  using org.omg.dds.core.policy;
4  using org.omg.dds.core.policy.modifiable;
5  using System;
6  using System.Collections.Generic;
7  using System.Linq;
8  using System.Text;
9  using System.Threading.Tasks;
10
11 namespace Doopec.DDS.Core.Policy
12 {
13     public class EntityFactoryQosPolicyImpl : QosPolicyImpl, EntityFactoryQosPolicy
14     {
15
16         public bool AutoenableCreatedEntities { get; protected internal set; }
17
18         public EntityFactoryQosPolicyImpl(Bootstrap boostrap)
19             : base(boostrap)
20         {
21         }
22
23         public EntityFactoryQosPolicyImpl(bool isAutoEnable, Bootstrap boostrap)
24             : base(boostrap)
25         {
26             this.AutoenableCreatedEntities  = isAutoEnable;
27         }
28
29         public bool IsAutoEnableCreatedEntities()
30         {
31             return AutoenableCreatedEntities;
32         }
33
34         public ModifiableEntityFactoryQosPolicy Modify()
35         {
36             return new ModifiableEntityFactoryQosPolicyImpl(this);
37         }
38     }
39 }
40
```

```
1  using Doopec.Dds.Core.Policy.modifiable;
2  using Doopec.DDS.Core.Policy;
3  using org.omg.dds.core;
4  using org.omg.dds.core.policy;
5  using org.omg.dds.core.policy.modifiable;
6  using System;
7
8  namespace Doopec.Dds.Core.Policy
9  {
10     public class GroupDataQosPolicyImpl : QosPolicyImpl, GroupDataQosPolicy
11     {
12         public byte[] Value { get; protected internal set; }
13
14         public GroupDataQosPolicyImpl(Bootstrap boostrap)
15             : base(boostrap)
16         {
17         }
18         public GroupDataQosPolicyImpl(byte[] value, Bootstrap boostrap)
19             : base(boostrap)
20         {
21             this.Value  = value;
22         }
23
24         public int GetValue(byte[] value, int offset)
25         {
26             Array.Copy(Value, 0, value, offset, Value.Length - offset);
27
28             return Value.Length;
29         }
30
31         public int GetLength()
32         {
33
34             return Value.Length;
35         }
36
37         public ModifiableGroupDataQosPolicy Modify()
38         {
39             return new ModifiableGroupDataQosPolicyImpl(this);
40         }
41     }
42 }
43
```

```
using Doopec.Dds.Core.Policy.modifiable;
using Doopec.DDS.Core.Policy;
using org.omg.dds.core;
using org.omg.dds.core.policy;
using org.omg.dds.core.policy.modifiable;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Doopec.Dds.Core.Policy
{
    public class HistoryQosPolicyImpl : QosPolicyImpl, HistoryQosPolicy
    {



        public HistoryQosPolicyKind KindQos { get; protected internal set; }
        public int DepthQos { get; protected internal set; }


         public HistoryQosPolicyImpl(Bootstrap boostrap)
            : base(boostrap)
        {
        }


        public HistoryQosPolicyImpl(HistoryQosPolicyKind kind, int getDepth,Bootstrap boostrap)
            : base(boostrap)
        {
            this.KindQos  = kind;
            this.DepthQos = getDepth;
        }

        public HistoryQosPolicyKind GetKind()
        {
            return KindQos;
        }

        public int GetDepth()
        {
            return DepthQos ;
        }


        public ModifiableHistoryQosPolicy Modify()
        {
            return new ModifiableHistoryQosPolicyImpl(this);
        }
    }
}

```

```
using Doopec.Dds.Core.Policy.modifiable;
using Doopec.DDS.Core.Policy;
using org.omg.dds.core;
using org.omg.dds.core.policy;
using org.omg.dds.core.policy.modifiable;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Doopec.Dds.Core.Policy
{
    public class LatencyBudgetQosPolicyImpl : QosPolicyImpl, LatencyBudgetQosPolicy
    {


        public Duration GetDurationQos { get; protected internal set; }

         public LatencyBudgetQosPolicyImpl(Bootstrap boostrap)
            : base(boostrap)
        {
        }




        public LatencyBudgetQosPolicyImpl(Duration getDuration, Bootstrap boostrap)
            : base(boostrap)
        {
            this.GetDurationQos= getDuration;

        }
        public Duration GetDuration()
        {
            return GetDurationQos;
        }



        public ModifiableLatencyBudgetQosPolicy Modify()
        {
            return new ModifiableLatencyBudgetQosPolicyImpl(this);
        }
    }
}

```

```
using Doopec.Dds.Core.Policy.modifiable;
using Doopec.DDS.Core.Policy;
using org.omg.dds.core;
using org.omg.dds.core.policy;
using org.omg.dds.core.policy.modifiable;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Doopec.Dds.Core.Policy
{
    public class LifespanQosPolicyImpl : QosPolicyImpl, LifespanQosPolicy
    {


        public Duration GetDurationQos { get; protected internal set; }


         public LifespanQosPolicyImpl(Bootstrap boostrap)
            : base(boostrap)
        {
        }
        public LifespanQosPolicyImpl(Duration getDuration,Bootstrap boostrap)
            :base(boostrap)
        {
            this.GetDurationQos = getDuration;

        }
        public Duration GetDuration()
        {
            return GetDurationQos;
        }



        public ModifiableLifespanQosPolicy Modify()
        {
            return new ModifiableLifespanQosPolicyImpl(this);
        }
    }
}

```

```
using Doopec.Dds.Core.Policy.modifiable;
using Doopec.DDS.Core.Policy;
using org.omg.dds.core;
using org.omg.dds.core.policy;
using org.omg.dds.core.policy.modifiable;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Doopec.Dds.Core.Policy
{
    public class LivelinessQosPolicyImpl : QosPolicyImpl,LivelinessQosPolicy
    {

        public LivelinessQosPolicyKind KindQos { get; protected internal set; }
        public Duration LeaseDurationQos { get; protected internal set; }

        public LivelinessQosPolicyImpl(Bootstrap boostrap)
            : base(boostrap)
        {
        }
        public LivelinessQosPolicyImpl(LivelinessQosPolicyKind kind, Duration getLeaseDuration, Bootstrap ↙
    boostrap)
            :base(boostrap)
        {
            this.KindQos  = kind;
            this.LeaseDurationQos = getLeaseDuration;

        }

        public LivelinessQosPolicyImpl(LivelinessQosPolicyKind kind, Bootstrap boostrap)
            : this(boostrap)
        {
            this.KindQos = kind;
        }
        public LivelinessQosPolicyKind GetKind()
        {
            return KindQos ;
        }

        public Duration GetLeaseDuration()
        {
            return LeaseDurationQos;
        }



        public ModifiableLivelinessQosPolicy Modify()
        {
            return new ModifiableLivelinessQosPolicyImpl(this);
        }
    }
}

```

```
1  using Doopec.Dds.Core.Policy.modifiable;
2  using Doopec.DDS.Core.Policy;
3  using org.omg.dds.core;
4  using org.omg.dds.core.policy;
5  using org.omg.dds.core.policy.modifiable;
6  using System;
7  using System.Collections.Generic;
8  using System.Linq;
9  using System.Text;
10 using System.Threading.Tasks;
11
12 namespace Doopec.Dds.Core.Policy
13 {
14     public class OwnershipQosPolicyImpl : QosPolicyImpl, OwnershipQosPolicy
15     {
16
17         public OwnershipQosPolicyKind KindQos { get; protected internal set; }
18
19         public OwnershipQosPolicyImpl(Bootstrap boostrap)
20             : base(boostrap)
21         {
22
23         }
24         public OwnershipQosPolicyImpl(OwnershipQosPolicyKind kind, Bootstrap boostrap)
25             :base(boostrap)
26         {
27             this.KindQos = kind;
28         }
29         public OwnershipQosPolicyKind GetKind()
30         {
31             return KindQos;
32         }
33
34
35
36         public ModifiableOwnershipQosPolicy Modify()
37         {
38             return new ModifiableOwnershipQosPolicyImpl(this);
39         }
40     }
41 }
42
```

```
1  using Doopec.Dds.Core.Policy.modifiable;
2  using Doopec.DDS.Core.Policy;
3  using org.omg.dds.core;
4  using org.omg.dds.core.policy;
5  using org.omg.dds.core.policy.modifiable;
6  using System;
7  using System.Collections.Generic;
8  using System.Linq;
9  using System.Text;
10 using System.Threading.Tasks;
11
12
13 namespace Doopec.Dds.Core.Policy
14 {
15     public class OwnershipStrengthQosPolicyImpl : QosPolicyImpl, OwnershipStrengthQosPolicy
16     {
17         public int StrengthQos { get; protected internal set; }
18
19
20         public OwnershipStrengthQosPolicyImpl(Bootstrap boostrap)
21             : base(boostrap)
22         {
23
24         }
25         public OwnershipStrengthQosPolicyImpl(int strength,Bootstrap boostrap)
26             :base(boostrap)
27         {
28             this.StrengthQos = strength;
29         }
30
31         public int GetValue()
32         {
33             return StrengthQos;
34         }
35
36
37
38         public ModifiableOwnershipStrengthQosPolicy Modify()
39         {
40             return new ModifiableOwnershipStrengthQosPolicyImpl(this);
41         }
42     }
43 }
44
```

```
1  using Doopec.Dds.Core.Policy.modifiable;
2  using Doopec.DDS.Core.Policy;
3  using org.omg.dds.core;
4  using org.omg.dds.core.policy;
5  using org.omg.dds.core.policy.modifiable;
6  using System;
7  using System.Collections.Generic;
8
9  namespace Doopec.Dds.Core.Policy
10 {
11     public class PartitionQosPolicyImpl : QosPolicyImpl, PartitionQosPolicy
12     {
13
14         public ICollection<string> NameQos { get; protected internal set; }
15         public PartitionQosPolicyImpl(Bootstrap boostrap)
16             : base(boostrap)
17         {
18         }
19
20         public PartitionQosPolicyImpl(ICollection<string> getName, Bootstrap boostrap)
21             : base(boostrap)
22         {
23             this.NameQos = getName;
24         }
25         public ICollection<string> GetName()
26         {
27             return NameQos;
28         }
29
30         public ModifiablePartitionQosPolicy Modify()
31         {
32             return new ModifiablePartitionQosPolicyImpl(this);
33         }
34     }
35 }
36
```

```
1  using Doopec.Dds.Core.Policy.modifiable;
2  using Doopec.DDS.Core.Policy;
3  using org.omg.dds.core;
4  using org.omg.dds.core.policy;
5  using org.omg.dds.core.policy.modifiable;
6  using System;
7  using System.Collections.Generic;
8  using System.Linq;
9  using System.Text;
10 using System.Threading.Tasks;
11
12 namespace Doopec.Dds.Core.Policy
13 {
14     public class PresentationQosPolicyImpl : QosPolicyImpl, PresentationQosPolicy
15     {
16         public AccessScopeKind AccessScopeKindQos { get; protected internal set; }
17
18
19         public bool IsCoherentAccessQos { get; protected internal set; }
20
21
22         public bool IsOrderedAccessQos { get; protected internal set; }
23
24         public PresentationQosPolicyImpl(Bootstrap boostrap)
25             : base(boostrap)
26         {
27         }
28         public PresentationQosPolicyImpl(AccessScopeKind kind,bool isCoherentAccess,bool isOrderedAccess, ↙
       Bootstrap boostrap)
29             : base(boostrap)
30         {
31             this.AccessScopeKindQos = kind;
32             this.IsCoherentAccessQos = isCoherentAccess;
33             this.IsOrderedAccessQos = isOrderedAccess;
34         }
35         public AccessScopeKind GetAccessScope()
36         {
37             return AccessScopeKindQos;
38         }
39
40         public bool IsCoherentAccess()
41         {
42             return IsCoherentAccessQos;
43         }
44
45         public bool IsOrderedAccess()
46         {
47             return IsOrderedAccessQos;
48         }
49
50         public ModifiablePresentationQosPolicy Modify()
51         {
52             return new ModifiablePresentationQosPolicyImpl(this);
53         }
54     }
55 }
56
```

```
1  using org.omg.dds.core;
2  using org.omg.dds.core.policy;
3  using System;
4  using System.Collections.Generic;
5  using System.Linq;
6  using System.Text;
7  using System.Threading.Tasks;
8
9  namespace Doopec.Dds.Core.Policy
10 {
11     public class QosPolicyCountImpl : QosPolicy, QosPolicyCount
12     {
13         public QosPolicyId PolicyIdQos{ get; protected internal set; }
14
15         public int CountQos{ get; protected internal set; }
16
17
18         public QosPolicyCountImpl(Bootstrap boostrap)
19             : base()
20         {
21         }
22
23         public QosPolicyCountImpl(QosPolicyId getPolicyId, int getCount)
24             : base()
25         {
26             this.PolicyIdQos = getPolicyId;
27             this.CountQos = getCount;
28         }
29
30
31
32         public QosPolicyId GetPolicyId()
33         {
34             return PolicyIdQos;
35         }
36
37         public int GetCount()
38         {
39             return CountQos;
40         }
41         /// <summary>
42         /// To Review
43         /// </summary>
44         /// <returns>No se implemento automaticamente pero fue necesario para no tener errores en la
       compilacion</returns>
45         public QosPolicyId GetId()
46         {
47             throw new NotImplementedException();
48         }
49         public QosPolicyCount CopyFrom(QosPolicyCount other)
50         {
51             return new QosPolicyCountImpl(other.GetPolicyId(),other.GetCount());
52         }
53
54         public QosPolicyCount FinishModification()
55         {
56             return new QosPolicyCountImpl (this.GetPolicyId(),this.GetCount());
57         }
58
59         public QosPolicyCount Modify()
60         {
61             return new QosPolicyCountImpl(this.GetPolicyId(), this.GetCount());
62         }
63
64         public Bootstrap GetBootstrap()
65         {
66             throw new NotImplementedException();
67         }
68     }
69 }
70
```

```
1  using org.omg.dds.core;
2  using org.omg.dds.core.policy;
3  using System;
4
5  namespace Doopec.DDS.Core.Policy
6  {
7      public class QosPolicyImpl : QosPolicy
8      {
9          public Bootstrap Boostrap { get; internal set; }
10
11         public QosPolicyImpl(Bootstrap boostrap)
12         {
13             Boostrap = boostrap;
14         }
15
16         public QosPolicyId GetId()
17         {
18             throw new NotImplementedException();
19         }
20
21         public  Bootstrap GetBootstrap()
22         {
23             return Boostrap;
24         }
25     }
26
27     public class QosPolicyIdImpl : QosPolicyId
28     {
29         protected int policyIdValue;
30         protected string policyName;
31
32         public override int GetPolicyIdValue()
33         {
34             return policyIdValue;
35         }
36
37         public override string GetPolicyName()
38         {
39             return policyName;
40         }
41     }
42 }
43
```

```
1  using Doopec.Dds.Core.Policy.modifiable;
2  using Doopec.DDS.Core.Policy;
3  using org.omg.dds.core;
4  using org.omg.dds.core.policy;
5  using org.omg.dds.core.policy.modifiable;
6  using System;
7  using System.Collections.Generic;
8  using System.Linq;
9  using System.Text;
10 using System.Threading.Tasks;
11
12 namespace Doopec.Dds.Core.Policy
13 {
14     public class ReaderDataLifecycleQosPolicyImpl : QosPolicyImpl,ReaderDataLifecycleQosPolicy
15     {
16         public Duration AutoPurgeNoWriterSamplesDelay { get; protected internal set; }
17         public Duration AutoPurgeDisposedSamplesDelay { get; protected internal set; }
18         public ReaderDataLifecycleQosPolicyImpl(Bootstrap boostrap)
19             : base(boostrap)
20         {
21         }
22
23         public ReaderDataLifecycleQosPolicyImpl(Duration getAutoPurgeNoWriterSamplesDelay , Duration
       getAutoPurgeDisposedSamplesDelay,Bootstrap boostrap )
24             :base(boostrap)
25         {
26             this.AutoPurgeNoWriterSamplesDelay = getAutoPurgeNoWriterSamplesDelay;
27             this.AutoPurgeDisposedSamplesDelay = getAutoPurgeDisposedSamplesDelay;
28         }
29         public Duration GetAutoPurgeNoWriterSamplesDelay()
30         {
31             return AutoPurgeNoWriterSamplesDelay;
32         }
33
34         public Duration GetAutoPurgeDisposedSamplesDelay()
35         {
36             return AutoPurgeDisposedSamplesDelay;
37         }
38
39
40
41         public ModifiableReaderDataLifecycleQosPolicy Modify()
42         {
43             return new ModifiableReaderDataLifecycleQosPolicyImpl(this);
44         }
45     }
46 }
47
```

```
using Doopec.Dds.Core.Policy.modifiable;
using Doopec.DDS.Core.Policy;
using org.omg.dds.core;
using org.omg.dds.core.policy;
using org.omg.dds.core.policy.modifiable;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Doopec.Dds.Core.Policy
{
    public class ReliabilityQosPolicyImpl : QosPolicyImpl, ReliabilityQosPolicy
    {
        public ReliabilityQosPolicyKind KindQos { get; protected internal set; }
        public Duration MaxBlockingTimeQos { get; protected internal set; }

        public ReliabilityQosPolicyImpl( Bootstrap boostrap)
            : base(boostrap)
        {
            MaxBlockingTimeQos = Duration.ZeroDuration(boostrap);
        }

        public ReliabilityQosPolicyImpl(ReliabilityQosPolicyKind kind, Duration maxBlockingTime, Bootstrap
     boostrap)
            : this(boostrap)
        {
            this.KindQos = kind;
            this.MaxBlockingTimeQos = maxBlockingTime;
        }

        public ReliabilityQosPolicyImpl(ReliabilityQosPolicyKind kind, Bootstrap boostrap)
            : this(boostrap)
        {
            this.KindQos = kind;
        }

        public ReliabilityQosPolicyKind GetKind()
        {
            return KindQos;
        }

        public Duration GetMaxBlockingTime()
        {
            return MaxBlockingTimeQos;
        }

        public ModifiableReliabilityQosPolicy Modify()
        {
            return new ModifiableReliabilityQosPolicyImpl(this);
        }
    }



    /*
    public class ReliabilityQosPolicyImpl : QosPolicyImpl, ReliabilityQosPolicy
    {
        private readonly ReliabilityQosPolicyKind kind = ReliabilityQosPolicyKind.RELIABLE;
        private readonly Duration maxBlockingTime;

        public ReliabilityQosPolicyImpl(Bootstrap boostrap)
            : base(boostrap)
        {
            maxBlockingTime = Duration.ZeroDuration(boostrap);
        }

        public ReliabilityQosPolicyImpl(ReliabilityQosPolicyKind kind, Duration maxBlockingTime, Bootstrap
     boostrap)
            : this(boostrap)
        {
            this.kind = kind;
            this.maxBlockingTime = maxBlockingTime;
```

```
73          }
74
75          public ReliabilityQosPolicyImpl(ReliabilityQosPolicyKind kind, Bootstrap boostrap)
76              : this(boostrap)
77          {
78              this.kind = kind;
79          }
80
81          public ReliabilityQosPolicyKind GetKind()
82          {
83              return kind;
84          }
85
86          public Duration GetMaxBlockingTime()
87          {
88              return maxBlockingTime;
89          }
90
91          public ModifiableReliabilityQosPolicy Modify()
92          {
93              throw new NotImplementedException();
94          }
95      }*/
96  }
97
```

```
using Doopec.Dds.Core.Policy.modifiable;
using Doopec.DDS.Core.Policy;
using org.omg.dds.core;
using org.omg.dds.core.policy;
using org.omg.dds.core.policy.modifiable;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Doopec.Dds.Core.Policy
{
    public class ResourceLimitsQosPolicyImpl : QosPolicyImpl, ResourceLimitsQosPolicy
    {
        public int MaxSamplesQos { get; protected internal set; }

        public int MaxInstancesQos { get; protected internal set; }

        public int MaxSamplesPerInstanceQos { get; protected internal set; }
        public ResourceLimitsQosPolicyImpl(Bootstrap boostrap)
            : base(boostrap)
        {
        }
        public ResourceLimitsQosPolicyImpl (int getMaxSamples, int getMaxInstances, int getMaxSamplesPerInstance,Bootstrap boostrap)
            :base(boostrap)
        {
            this.MaxSamplesQos = getMaxSamples;
            this.MaxInstancesQos = getMaxInstances;
            this.MaxSamplesPerInstanceQos = getMaxSamplesPerInstance;
        }
        public int GetMaxSamples()
        {
            return MaxSamplesQos;
        }

        public int GetMaxInstances()
        {
            return MaxInstancesQos;
        }

        public int GetMaxSamplesPerInstance()
        {
            return MaxSamplesPerInstanceQos;
        }


        public ModifiableResourceLimitsQosPolicy Modify()
        {
            return new ModifiableResourceLimitsQosPolicyImpl(this);
        }
    }
}

```

```
1  using Doopec.Dds.Core.Policy.modifiable;
2  using Doopec.DDS.Core.Policy;
3  using org.omg.dds.core;
4  using org.omg.dds.core.policy;
5  using org.omg.dds.core.policy.modifiable;
6  using System;
7  using System.Collections.Generic;
8  using System.Linq;
9  using System.Text;
10 using System.Threading.Tasks;
11
12 namespace Doopec.Dds.Core.Policy
13 {
14     public class TimeBasedFilterQosPolicyImpl : QosPolicyImpl, TimeBasedFilterQosPolicy
15     {
16         public Duration MinimumSeparationQos { get; protected internal set; }
17         public TimeBasedFilterQosPolicyImpl(Bootstrap boostrap)
18             : base(boostrap)
19         {
20         }
21         public TimeBasedFilterQosPolicyImpl(Duration getMinimumSeparation,Bootstrap boostrap)
22             :base(boostrap)
23         {
24             this.MinimumSeparationQos = getMinimumSeparation;
25         }
26         public Duration GetMinimumSeparation()
27         {
28             return MinimumSeparationQos;
29         }
30
31
32         public ModifiableTimeBasedFilterQosPolicy Modify()
33         {
34             return new ModifiableTimeBasedFilterQosPolicyImpl(this);
35         }
36     }
37 }
38
```

```
1  using Doopec.Dds.Core.Policy.modifiable;
2  using Doopec.DDS.Core.Policy;
3  using org.omg.dds.core;
4  using org.omg.dds.core.policy;
5  using org.omg.dds.core.policy.modifiable;
6  using System;
7  using System.Collections.Generic;
8  using System.Linq;
9  using System.Text;
10 using System.Threading.Tasks;
11
12 namespace Doopec.Dds.Core.Policy
13 {
14     public class TopicDataQosPolicyImpl : QosPolicyImpl,TopicDataQosPolicy
15     {
16         public byte[] ValueQos { get; protected internal set; }
17
18         public TopicDataQosPolicyImpl(Bootstrap boostrap)
19             :base(boostrap)
20         {
21             this.ValueQos = new byte[0];
22         }
23
24         public TopicDataQosPolicyImpl(byte[] value, Bootstrap boostrap)
25             :base(boostrap)
26         {
27             this.ValueQos = value;
28         }
29
30         public int GetValue(byte[] result, int offset)
31         {
32             int length = this.ValueQos.Length;
33             Array.Copy(this.ValueQos, offset, result, 0, length);
34             return offset + length;
35         }
36
37         public int GetLength()
38         {
39             return this.ValueQos.Length;
40         }
41
42
43         public ModifiableTopicDataQosPolicy Modify()
44         {
45             return new ModifiableTopicDataQosPolicyImpl(this);
46         }
47     }
48 }
49
```

```
1  using Doopec.Dds.Core.Policy.modifiable;
2  using Doopec.DDS.Core.Policy;
3  using org.omg.dds.core;
4  using org.omg.dds.core.policy;
5  using org.omg.dds.core.policy.modifiable;
6  using System;
7  using System.Collections.Generic;
8  using System.Linq;
9  using System.Text;
10 using System.Threading.Tasks;
11
12 namespace Doopec.Dds.Core.Policy
13 {
14     class TransportPriorityQosPolicyImpl : QosPolicyImpl,TransportPriorityQosPolicy
15     {
16         public int ValueQos { get; protected internal set; }
17
18         public TransportPriorityQosPolicyImpl(Bootstrap boostrap)
19             : base(boostrap)
20         {
21         }
22         public TransportPriorityQosPolicyImpl(int getValue, Bootstrap boostrap)
23             :base(boostrap)
24         {
25             this.ValueQos = getValue;
26         }
27
28         public int GetValue()
29         {
30             return ValueQos;
31         }
32
33
34         public ModifiableTransportPriorityQosPolicy Modify()
35         {
36             return new ModifiableTransportPriorityQosPolicyImpl(this) ;
37         }
38     }
39 }
40
```

```
1  using Doopec.Dds.Core.Policy.modifiable;
2  using Doopec.DDS.Core.Policy;
3  using org.omg.dds.core;
4  using org.omg.dds.core.policy;
5  using org.omg.dds.core.policy.modifiable;
6  using System;
7  using System.Collections.Generic;
8  using System.Linq;
9  using System.Text;
10 using System.Threading.Tasks;
11
12 namespace Doopec.Dds.Core.Policy
13 {
14     public class TypeConsistencyEnforcementQosPolicyImpl: QosPolicyImpl,
       TypeConsistencyEnforcementQosPolicy
15     {
16         public TypeConsistencyEnforcementQosPolicyKind KindQos { get; protected internal set; }
17
18         public TypeConsistencyEnforcementQosPolicyImpl(TypeConsistencyEnforcementQosPolicyKind getKind,
       Bootstrap boostrap)
19             :base(boostrap)
20         {
21             this.KindQos = getKind;
22         }
23         public TypeConsistencyEnforcementQosPolicyKind GetKind()
24         {
25             return KindQos;
26         }
27
28
29         public ModifiableTypeConsistencyEnforcementQosPolicy Modify()
30         {
31             return new ModifiableTypeConsistencyEnforcementQosPolicyImpl(this);
32         }
33     }
34 }
35
```

```
using Doopec.Dds.Core.Policy.modifiable;
using org.omg.dds.core;
using org.omg.dds.core.policy;
using org.omg.dds.core.policy.modifiable;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Doopec.DDS.Core.Policy
{
    public class UserDataQosPolicyImpl : QosPolicyImpl, UserDataQosPolicy
    {
        public byte[] ValueQos { get; protected internal set; }

        public UserDataQosPolicyImpl(Bootstrap boostrap)
            : base(boostrap)
        {
            this.ValueQos = new byte[0];
        }

        public UserDataQosPolicyImpl(byte[] value, Bootstrap boostrap)
            : base(boostrap)
        {
            this.ValueQos = value;
        }

        public int GetValue(byte[] result, int offset)
        {
            int length = this.ValueQos.Length;
            Array.Copy(this.ValueQos, offset, result, 0, length);
            return offset + length;
        }

        public int GetLength()
        {
            return this.ValueQos.Length;
        }

        public ModifiableUserDataQosPolicy Modify()
        {
            return new ModifiableUserDataQosPolicyImpl(this);
        }
    }
}

```

```
1  using Doopec.Dds.Core.Policy.modifiable;
2  using Doopec.DDS.Core.Policy;
3  using org.omg.dds.core;
4  using org.omg.dds.core.policy;
5  using org.omg.dds.core.policy.modifiable;
6  using System;
7  using System.Collections.Generic;
8  using System.Linq;
9  using System.Text;
10 using System.Threading.Tasks;
11
12 namespace Doopec.Dds.Core.Policy
13 {
14     public class WriterDataLifecycleQosPolicyImpl : QosPolicyImpl, WriterDataLifecycleQosPolicy
15     {
16         public bool IsAutDisposeUnregisteredInstancesQos { get; protected internal set; }
17
18         public WriterDataLifecycleQosPolicyImpl(Bootstrap boostrap)
19             : base(boostrap)
20         {
21         }
22         public WriterDataLifecycleQosPolicyImpl(bool isAutDisposeUnregisteredInstancesQos, Bootstrap
       boostrap)
23             :base(boostrap)
24         {
25             this.IsAutDisposeUnregisteredInstancesQos =isAutDisposeUnregisteredInstancesQos;
26         }
27
28         public bool IsAutDisposeUnregisteredInstances()
29         {
30             return IsAutDisposeUnregisteredInstancesQos;
31         }
32
33         public ModifiableWriterDataLifecycleQosPolicy Modify()
34         {
35             return new ModifiableWriterDataLifecycleQosPolicyImpl(this);
36         }
37     }
38 }
39
```

```
1
2  using DDS.ConversionUtils;
3  using Doopec.Dds.Domain;
4  using Doopec.Dds.Utils;
5  using org.omg.dds.core;
6  using org.omg.dds.core.modifiable;
7  using org.omg.dds.domain;
8  using org.omg.dds.topic;
9  using org.omg.dds.type;
10 using org.omg.dds.type.dynamic;
11 using System;
12 using System.Collections.Generic;
13
14 namespace Doopec.Dds.Core
15 {
16
17     /// <summary>
18     /// A Bootstrap object represents an instantiation of a Service implementation within a process. It
19     /// is the “root” for all other DDS objects and assists in their creation by means of an internal
20     /// service-provider interface. All stateful types in this PSM implement an interface DDSObject,
21     /// through a getBootstrap method on which they can provide access to the Bootstrap from
22     /// which they are ultimately derived. (Bootstrap itself implements this interface; a Bootstrap
23     /// always returns this from its getBootstrap operation.)
24     /// The Bootstrap class allows implementations to avoid the presence of static state, if desired. It
25     /// also allows multiple DDS implementations—or multiple versions of the “same”
26     /// implementation—to potentially coexist within the same Java run-time environment. A DDS
27     /// application’s first step is to instantiate a Bootstrap, which represents the DDS implementation
28     /// that it will use. From there, it can create all of its additional DDS objects.
29     /// The Bootstrap class is abstract. To avoid compile-time dependencies on concrete
30     /// Bootstrap implementations, an application can instantiate a Bootstrap by means of a static
31     /// createInstance method on the Bootstrap class. This method looks up a concrete
32     /// Bootstrap subclass using a Java system property containing the name of that subclass. This
33     /// subclass must be provided by implementers and will therefore have an implementation-specific
34     /// name.
35     /// </summary>
36     public class BootstrapImpl : Bootstrap
37     {
38         private SPI SPIInstance;
39
40         static BootstrapImpl()
41         {
42             // Activate the Discovery Service
43             var dicovery = DiscoveryService.Instance;
44         }
45
46         public BootstrapImpl()
47             : this(null)
48         { }
49
50         public BootstrapImpl(IDictionary<string, Object> environment)
51         { }
52
53         public override Bootstrap.ServiceProviderInterface GetSPI()
54         {
55             if (SPIInstance == null)
56                 SPIInstance = new SPI(this);
57
58             return SPIInstance;
59         }
60     }
61
62     public class SPI : Bootstrap.ServiceProviderInterface
63     {
64
65         private readonly Bootstrap boostrap;
66
67         public SPI(Bootstrap boostrap)
68         {
69             this.boostrap = boostrap;
70         }
71
72         public DomainParticipantFactory GetParticipantFactory()
73         {
74             return new DomainParticipantFactoryImpl(this.boostrap);
```

```
        }

        public DynamicTypeFactory GetTypeFactory()
        {
            throw new NotImplementedException();
        }

        public DynamicDataFactory GetDataFactory()
        {
            throw new NotImplementedException();
        }

        public TypeSupport<TYPE> NewTypeSupport<TYPE>(Type type, string registeredName)
        {
            throw new NotImplementedException();
        }

        public ModifiableDuration NewDuration(long duration, TimeUnit unit)
        {
            throw new NotImplementedException();
        }

        public Duration InfiniteDuration()
        {
            return new DurationImpl(this.boostrap, long.MaxValue);
        }

        public Duration ZeroDuration()
        {
            return new DurationImpl(this.boostrap, 0);
        }

        public ModifiableTime NewTime(long time, TimeUnit units)
        {
            throw new NotImplementedException();
        }

        public Time InvalidTime()
        {
            throw new NotImplementedException();
        }

        public ModifiableInstanceHandle NewInstanceHandle()
        {
            throw new NotImplementedException();
        }

        public InstanceHandle NilHandle()
        {
            throw new NotImplementedException();
        }

        public GuardCondition NewGuardCondition()
        {
            throw new NotImplementedException();
        }

        public WaitSet NewWaitSet()
        {
            throw new NotImplementedException();
        }

        public BuiltinTopicKey NewBuiltinTopicKey()
        {
            throw new NotImplementedException();
        }

        public org.omg.dds.topic.ParticipantBuiltinTopicData NewParticipantBuiltinTopicData()
        {
            throw new NotImplementedException();
        }

        public org.omg.dds.topic.PublicationBuiltinTopicData NewPublicationBuiltinTopicData()
        {
```

```
149             throw new NotImplementedException();
150         }
151 
152         public org.omg.dds.topic.SubscriptionBuiltinTopicData NewSubscriptionBuiltinTopicData()
153         {
154             throw new NotImplementedException();
155         }
156 
157         public org.omg.dds.topic.TopicBuiltinTopicData NewTopicBuiltinTopicData()
158         {
159             throw new NotImplementedException();
160         }
161 
162         public org.omg.dds.core.policy.QosPolicyId GetQosPolicyId(Type policyClass)
163         {
164             throw new NotImplementedException();
165         }
166 
167         public ISet<Type> AllStatusKinds()
168         {
169             throw new NotImplementedException();
170         }
171 
172         public ISet<Type> NoStatusKinds()
173         {
174             throw new NotImplementedException();
175         }
176 
177         public org.omg.dds.core.status.LivelinessLostStatus<TYPE> NewLivelinessLostStatus<TYPE>()
178         {
179             throw new NotImplementedException();
180         }
181 
182         public org.omg.dds.core.status.OfferedDeadlineMissedStatus<TYPE> NewOfferedDeadlineMissedStatus ↙
    <TYPE>()
183         {
184             throw new NotImplementedException();
185         }
186 
187         public org.omg.dds.core.status.OfferedIncompatibleQosStatus<TYPE> NewOfferedIncompatibleQosStatus↙
    <TYPE>()
188         {
189             throw new NotImplementedException();
190         }
191 
192         public org.omg.dds.core.status.PublicationMatchedStatus<TYPE> NewPublicationMatchedStatus<TYPE>()
193         {
194             throw new NotImplementedException();
195         }
196 
197         public org.omg.dds.core.status.LivelinessChangedStatus<TYPE> NewLivelinessChangedStatus<TYPE>()
198         {
199             throw new NotImplementedException();
200         }
201 
202         public org.omg.dds.core.status.RequestedDeadlineMissedStatus<TYPE>                              ↙
    NewRequestedDeadlineMissedStatus<TYPE>()
203         {
204             throw new NotImplementedException();
205         }
206 
207         public org.omg.dds.core.status.RequestedIncompatibleQosStatus<TYPE>                             ↙
    NewRequestedIncompatibleQosStatus<TYPE>()
208         {
209             throw new NotImplementedException();
210         }
211 
212         public org.omg.dds.core.status.SampleLostStatus<TYPE> NewSampleLostStatus<TYPE>()
213         {
214             throw new NotImplementedException();
215         }
216 
217         public org.omg.dds.core.status.SampleRejectedStatus<TYPE> NewSampleRejectedStatus<TYPE>()
218         {
```

```
219             throw new NotImplementedException();
220         }
221 
222         public org.omg.dds.core.status.SubscriptionMatchedStatus<TYPE> NewSubscriptionMatchedStatus<TYPE>↙
    ()
223         {
224             throw new NotImplementedException();
225         }
226 
227         public org.omg.dds.core.status.DataAvailableStatus<TYPE> NewDataAvailableStatus<TYPE>()
228         {
229             throw new NotImplementedException();
230         }
231 
232         public org.omg.dds.core.status.DataOnReadersStatus NewDataOnReadersStatus()
233         {
234             throw new NotImplementedException();
235         }
236 
237         public org.omg.dds.core.status.InconsistentTopicStatus<TYPE> NewInconsistentTopicStatus<TYPE>()
238         {
239             throw new NotImplementedException();
240         }
241 
242         public ISet<org.omg.dds.sub.InstanceState> AnyInstanceStateSet()
243         {
244             throw new NotImplementedException();
245         }
246 
247         public ISet<org.omg.dds.sub.InstanceState> NotAliveInstanceStateSet()
248         {
249             throw new NotImplementedException();
250         }
251 
252         public ISet<org.omg.dds.sub.SampleState> AnySampleStateSet()
253         {
254             throw new NotImplementedException();
255         }
256 
257         public ISet<org.omg.dds.sub.ViewState> AnyViewStateSet()
258         {
259             throw new NotImplementedException();
260         }
261     }
262 }
263 
```

```
using org.omg.dds.core.status;
using org.omg.dds.sub;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Doopec.DDS.Core
{
    public class DataAvailableStatusImpl<TYPE> : DataAvailableStatus<TYPE>
    {
        public DataAvailableStatusImpl(DataReader<TYPE> source)
            : base(source)
        {
        }

        public override DataReader<TYPE> GetSource()
        {
            return source as DataReader<TYPE>;
        }

        public override DataAvailableStatus<TYPE> CopyFrom(DataAvailableStatus<TYPE> other)
        {
            throw new NotImplementedException();
        }

        public override DataAvailableStatus<TYPE> FinishModification()
        {
            throw new NotImplementedException();
        }

        public override DataAvailableStatus<TYPE> Clone()
        {
            throw new NotImplementedException();
        }

        public override DataAvailableStatus<TYPE> Modify()
        {
            throw new NotImplementedException();
        }

        public override org.omg.dds.core.Bootstrap GetBootstrap()
        {
            throw new NotImplementedException();
        }
    }
}

```

```
using org.omg.dds.core;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Doopec.Dds.Core
{
    public class DDSObjectImpl : DDSObject
    {
        /// <summary>
        /// The Bootstrap object that directly or indirectly created this object.
        /// </summary>
        private readonly Bootstrap bootstrap;

        public DDSObjectImpl(Bootstrap bootstrap)
        {
            this.bootstrap = bootstrap;
        }

        /// <summary>
        /// The Bootstrap object that directly or indirectly created this object.
        /// </summary>
        public Bootstrap Bootstrap
        {
            get { return this.bootstrap; }
        }

        /// <summary>
        /// The Bootstrap object that directly or indirectly created this object.
        /// </summary>
         public Bootstrap GetBootstrap()
         {
             return this.bootstrap;
         }

    }
}

```

```
using DDS.ConversionUtils;
using org.omg.dds.core;
using org.omg.dds.core.modifiable;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Doopec.Dds.Core
{
    internal class DurationImpl : Duration
    {
        internal Bootstrap Boostrap { get; private set; }

        // in ticks values
        private TimeSpan durationValue;

        public DurationImpl(Bootstrap boostrap, long ticks)
        {
            this.Boostrap = boostrap;
            durationValue = new TimeSpan(ticks);
        }

        public override long GetDuration(TimeUnit inThisUnit)
        {
            switch (inThisUnit)
            {
                case TimeUnit.DAYS:
                    return (long)durationValue.TotalDays;
                case TimeUnit.HOURS:
                    return (long)durationValue.TotalHours;
                case TimeUnit.MICROSECONDS:
                    return (long)(durationValue.Ticks  / (TimeSpan.TicksPerMillisecond / 1000));
                case TimeUnit.MILLISECONDS:
                    return (long)durationValue.TotalMilliseconds;
                case TimeUnit.MINUTES:
                    return (long)durationValue.TotalMinutes;
                case TimeUnit.NANOSECONDS:
                    return (long)(durationValue.Ticks / (TimeSpan.TicksPerMillisecond / 1000000.0));
                case TimeUnit.SECONDS:
                    return (long)durationValue.TotalSeconds;

                default:
                    throw new ArgumentException("Invalid Time Unit");
            }
        }

        public double GetDurationInUnits(TimeUnit inThisUnit)
        {
            switch (inThisUnit)
            {
                case TimeUnit.DAYS:
                    return durationValue.TotalDays;
                case TimeUnit.HOURS:
                    return durationValue.TotalHours;
                case TimeUnit.MICROSECONDS:
                    return durationValue.Ticks / 1000.0;
                case TimeUnit.MILLISECONDS:
                    return durationValue.TotalMilliseconds;
                case TimeUnit.MINUTES:
                    return durationValue.TotalMinutes;
                case TimeUnit.NANOSECONDS:
                    return durationValue.Ticks;
                case TimeUnit.SECONDS:
                    return durationValue.TotalSeconds;

                default:
                    throw new ArgumentException("Invalid Time Unit");
            }
        }

        public override long GetRemainder(TimeUnit primaryUnit, TimeUnit remainderUnit)
        {
```

```
            double remainer = this.GetDurationInUnits(primaryUnit) - GetDuration(primaryUnit);

            throw new NotImplementedException();
        }

        public override bool IsZero()
        {
            return durationValue.Ticks == 0;
        }

        public override bool IsInfinite()
        {
            return durationValue.Ticks == long.MaxValue;
        }

        public override ModifiableDuration Modify()
        {
            throw new NotImplementedException();
        }

        public override Bootstrap GetBootstrap()
        {
            throw new NotImplementedException();
        }
    }
}

```

```
1  using DDS.ConversionUtils;
2  using org.omg.dds.core;
3  using System;
4  using System.Collections.Generic;
5  using System.Linq;
6  using System.Text;
7  using System.Threading.Tasks;
8
9  namespace Doopec.Dds.Core
10 {
11     public abstract class EntityImpl<SELF, LISTENER, QOS> : DDSObjectImpl, Entity<SELF, LISTENER, QOS>,
       IDisposable
12         where SELF : Entity<SELF, LISTENER, QOS>
13         where LISTENER : EventListener
14         where QOS : EntityQos
15     {
16         public LISTENER Listener { get; set; }
17         public QOS QoS { get; set; }
18         public bool IsEnabled { get; set; }
19
20         public bool IsClosed{ get; set; }
21
22         public StatusCondition<SELF> StatusCondition { get; set; }
23         public InstanceHandle InstanceHandle { get; set; }
24
25         public EntityImpl(Bootstrap bootstrap)
26             : base(bootstrap)
27         {
28             this.IsEnabled = false;
29             this.IsClosed = false;
30         }
31
32         public LISTENER GetListener()
33         {
34             return this.Listener;
35         }
36
37         public void SetListener(LISTENER listener)
38         {
39             this.Listener = listener;
40         }
41
42         public QOS GetQos()
43         {
44             return this.QoS;
45         }
46
47         public void SetQos(QOS qos)
48         {
49             this.QoS = qos;
50         }
51
52         public void SetQos(string qosLibraryName, string qosProfileName)
53         {
54             throw new NotImplementedException();
55         }
56
57         public virtual void Enable()
58         {
59             this.IsEnabled = true;
60         }
61
62         public StatusCondition<SELF> GetStatusCondition()
63         {
64             return this.StatusCondition;
65         }
66
67         public ICollection<TYPE> GetStatusChanges<TYPE>(ICollection<TYPE> statuses)
68         {
69             throw new NotImplementedException();
70         }
71
72         public InstanceHandle GetInstanceHandle()
73         {
```

```
74              return this.InstanceHandle;
75          }
76
77          public virtual void Close()
78          {
79              this.IsClosed = true;
80              this.IsEnabled = false;
81          }
82
83          public virtual void Retain()
84          {
85              throw new NotImplementedException();
86          }
87
88          public void Dispose()
89          {
90              this.Close();
91          }
92      }
93  }
94
```

```
1  using org.omg.dds.core;
2  using org.omg.dds.core.modifiable;
3  using System;
4
5  namespace Doopec.DDS.Core
6  {
7      public class ModifiableValueImpl<UNMOD_SELF, MOD_SELF> : ValueImpl<UNMOD_SELF, MOD_SELF>,
       ModifiableValue<UNMOD_SELF, MOD_SELF>
8          where UNMOD_SELF : Value<UNMOD_SELF, MOD_SELF>
9          where MOD_SELF : UNMOD_SELF
10     {
11         public MOD_SELF CopyFrom(UNMOD_SELF other)
12         {
13             throw new NotImplementedException();
14         }
15
16         public UNMOD_SELF FinishModification()
17         {
18             throw new NotImplementedException();
19         }
20     }
21 }
22
```

```
1  using org.omg.dds.core;
2  using System;
3
4  namespace Doopec.DDS.Core
5  {
6      public class ValueImpl<UNMOD_SELF, MOD_SELF> : Value<UNMOD_SELF, MOD_SELF>
7          where UNMOD_SELF : Value<UNMOD_SELF, MOD_SELF>
8          where MOD_SELF : UNMOD_SELF
9      {
10         public MOD_SELF Modify()
11         {
12             throw new NotImplementedException();
13         }
14
15         public Bootstrap GetBootstrap()
16         {
17             throw new NotImplementedException();
18         }
19     }
20 }
21
```

```
1  using Doopec.DDS.Core;
2  using org.omg.dds.core;
3  using org.omg.dds.core.policy;
4  using org.omg.dds.domain;
5  using org.omg.dds.domain.modifiable;
6  using System;
7  using System.Collections.Generic;
8  using System.Linq;
9  using System.Text;
10 using System.Threading.Tasks;
11
12 namespace Doopec.Dds.Domain
13 {
14     public class ModifiableDomainParticipantFactoryQosImpl : EntityQosImpl<DomainParticipantFactoryQos,
       ModifiableDomainParticipantFactoryQos>, ModifiableDomainParticipantFactoryQos
15     {
16         public ModifiableDomainParticipantFactoryQosImpl(DomainParticipantFactoryQosImpl qos)
17             : base(qos.GetBootstrap())
18         {
19
20         }
21         public ModifiableDomainParticipantFactoryQosImpl(Bootstrap boostrap)
22             : base(boostrap)
23         {
24
25         }
26
27         public override ModifiableDomainParticipantFactoryQos Modify()
28         {
29             throw new NotImplementedException();
30         }
31
32         public ModifiableDomainParticipantFactoryQos SetEntityFactory(EntityFactoryQosPolicy
       entityFactory)
33         {
34             throw new NotImplementedException();
35         }
36
37         public EntityFactoryQosPolicy GetEntityFactory()
38         {
39             throw new NotImplementedException();
40         }
41
42
43         public POLICY put<POLICY>(org.omg.dds.core.policy.QosPolicyId key, POLICY value) where POLICY :
       org.omg.dds.core.policy.QosPolicy
44         {
45             throw new NotImplementedException();
46         }
47
48         public ModifiableDomainParticipantFactoryQos CopyFrom(DomainParticipantFactoryQos other)
49         {
50             return new ModifiableDomainParticipantFactoryQosImpl(other.GetBootstrap());
51         }
52
53         public DomainParticipantFactoryQos FinishModification()
54         {
55             return new DomainParticipantFactoryQosImpl(this.GetBootstrap());
56         }
57     }
58 }
59
```

```
1  using Doopec.DDS.Domain;
2  using org.omg.dds.core;
3  using org.omg.dds.core.policy;
4  using org.omg.dds.domain;
5  using org.omg.dds.domain.modifiable;
6  using System;
7  using System.Collections.Generic;
8  using System.Linq;
9  using System.Text;
10 using System.Threading.Tasks;
11
12 namespace Doopec.Dds.Domain.modifiable
13 {
14     class ModifiableDomainParticipantQosImpl : DomainParticipantQosImpl, ModifiableDomainParticipantQos
15     {
16
17
18             public ModifiableDomainParticipantQosImpl(DomainParticipantQos dm)
19              : base(dm.GetUserData(),dm.GetEntityFactory(),dm.GetBootstrap())
20         {
21         }
22             public ModifiableDomainParticipantQosImpl(UserDataQosPolicy getUserData,EntityFactoryQosPolicy ↙
       getEntityFactory, Bootstrap boostrap)
23              : base(getUserData,getEntityFactory, boostrap )
24         {
25
26         }
27
28         public ModifiableDomainParticipantQos SetUserData(UserDataQosPolicy userData)
29         {
30             this.UserData=userData;
31             return this;
32         }
33
34         public ModifiableDomainParticipantQos SetEntityFactory(EntityFactoryQosPolicy entityFactory)
35         {
36             this.EntityFactoryQosPolicy =entityFactory;
37             return this;
38         }
39
40         public new System.Collections.IEnumerator GetEnumerator()
41         {
42             throw new NotImplementedException();
43         }
44
45         public POLICY put<POLICY>(QosPolicyId key, POLICY value) where POLICY : QosPolicy
46         {
47             throw new NotImplementedException();
48         }
49
50         public ModifiableDomainParticipantQos CopyFrom(DomainParticipantQos other)
51         {
52             return new ModifiableDomainParticipantQosImpl(other);
53         }
54
55         public DomainParticipantQos FinishModification()
56         {
57             return new DomainParticipantQosImpl(this.GetUserData(),this.GetEntityFactory(),this.          ↙
       GetBootstrap());
58         }
59     }
60 }
61
```

```
using Doopec.Configuration;
using Doopec.Dds.Core;
using Doopec.Dds.Utils;
using Doopec.DDS.Domain;
using Doopec.DDS.Utils;
using org.omg.dds.core;
using org.omg.dds.domain;
using System;
using System.Collections.Generic;
using System.Configuration;

namespace Doopec.Dds.Domain
{
    /// <summary>
    /// Implements the DomainParticipantFactory interfaces.
    ///
    /// This class acts as factory of the DomainParticipant.
    ///
    /// See the DDS specification, OMG formal/04-12-02, for a description of
    /// the interface this class is implementing.
    /// </summary>
    public class DomainParticipantFactoryImpl : DomainParticipantFactory
    {
        public DomainParticipantFactoryImpl(Bootstrap bootstrap)
        {
            this.bootstrap_ = bootstrap;
            string qosProfile = ddsConfig.Domains[0].QoSProfile.Name;
            Doopec.Configuration.Dds.DomainParticipantFactoryQoS qos = ddsConfig.QoSProfiles[qosProfile].↙
    DomainParticipantFactoryQos;

            this.Qos = DomainParticipantFactoryQosImpl.ConvertTo(qos, bootstrap);
        }

        /// <summary>
        /// Create a new participant in the domain with ID 0 having default QoS
        /// and no listener.
        /// </summary>
        /// <returns>New participant</returns>
        public override DomainParticipant CreateParticipant()
        {
            return this.CreateParticipant(0);
        }

        /// <summary>
        /// Create a new participant in the domain with a fixed ID and having default QoS
        /// and no listener.
        /// </summary>
        /// <returns>New participant</returns>
        public override DomainParticipant CreateParticipant(int domainId)
        {
            string qosProfile = ddsConfig.Domains[domainId].QoSProfile.Name;
            Doopec.Configuration.Dds.DomainParticipantQoS qosConfig = ddsConfig.QoSProfiles[qosProfile]. ↙
    DomainParticipantQos;

            DomainParticipantQos qos = DomainParticipantQosImpl.ConvertTo(qosConfig, this.bootstrap_);

            return this.CreateParticipant(domainId, qos, null, null);
        }

        /// <summary>
        /// Create a new domain participant.
        /// </summary>
        /// <param name="domainId"></param>
        /// <param name="qos"></param>
        /// <param name="listener"></param>
        /// <param name="statuses">Of which status changes the listener should be
        ///                        notified. A null collection signifies all status
        ///                        changes</param>
        /// <returns></returns>

        public override DomainParticipant CreateParticipant(int domainId, DomainParticipantQos qos,      ↙
    DomainParticipantListener listener, ICollection<Type> statuses)
        {
            DomainParticipant dp = new DomainParticipantImpl(domainId, qos, listener, this.bootstrap_);
```

```
72              if (((DomainParticipantFactoryQosImpl)this.Qos).EntityFactoryQosPolicy.
    IsAutoEnableCreatedEntities())
73                  dp.Enable();
74
75              return dp;
76          }
77
78          /// <summary>
79          /// Create a new domain participant.
80          /// </summary>
81          /// <param name="domainId"></param>
82          /// <param name="qosLibraryName"></param>
83          /// <param name="qosProfileName"></param>
84          /// <param name="listener"></param>
85          /// <param name="statuses">Of which status changes the listener should be
86          ///                        notified. A null collection signifies all status
87          ///                        changes</param>
88          /// <returns></returns>
89
90          public override DomainParticipant CreateParticipant(int domainId, string qosLibraryName, string
    qosProfileName, DomainParticipantListener listener, ICollection<Type> statuses)
91          {
92              throw new NotImplementedException();
93          }
94
95          /// <summary>
96          /// This operation retrieves a previously created DomainParticipant belonging to specified
    domain_id. If no such
97          /// DomainParticipant exists, the operation will return a 'nil' value.
98          /// If multiple DomainParticipant entities belonging to that domain_id exist, then the operation
    will return one of them. It is not
99          /// specified which one.
100         /// </summary>
101         /// <param name="domainId">The Domain identified by the domainId argument</param>
102         /// <returns></returns>
103         public override DomainParticipant LookupParticipant(int domainId)
104         {
105             DomainParticipant dp = DiscoveryService.Instance.LookupParticipant(domainId);
106             return dp;
107         }
108
109         public override DomainParticipantFactoryQos Qos
110         {
111             get { return qos_;  }
112             set
113             {
114                 if (QosHelper.IsValid(value) && QosHelper.IsConsistent(value))
115                 {
116                     if (!(qos_ == value) && QosHelper.IsChangeable(qos_, value))
117                         qos_ = value;
118                 }
119                 else
120                 {
121                     throw new InconsistentPolicyException();
122                 }
123             }
124
125         }
126
127
128         public override DomainParticipantQos GetDefaultParticipantQos()
129         {
130             return this.default_participant_qos_;
131         }
132
133         public override void SetDefaultParticipantQos(DomainParticipantQos qos)
134         {
135             if (QosHelper.IsValid(qos) && QosHelper.IsConsistent(qos))
136             {
137                 this.default_participant_qos_ = qos;
138             }
139             else
140             {
141                 throw new InconsistentPolicyException();
```

```
142             }
143         }
144 
145         public override void SetDefaultParticipantQos(string qosLibraryName, string qosProfileName)
146         {
147             throw new NotImplementedException();
148         }
149 
150         public override Bootstrap GetBootstrap()
151         {
152             return bootstrap_;
153         }
154 
155         #region Fields
156         private DomainParticipantFactoryQos qos_;
157 
158         /// The default qos Value of DomainParticipant.
159         private DomainParticipantQos default_participant_qos_;
160 
161 
162         private readonly Bootstrap bootstrap_ = null;
163 
164         private DDSConfigurationSection ddsConfig = Doopec.Configuration.DDSConfigurationSection.Instance↙
    ;
165 
166         #endregion
167     }
168 }
169 
```

```
1  using Doopec.DDS.Core;
2  using Doopec.DDS.Core.Policy;
3  using org.omg.dds.core;
4  using org.omg.dds.core.policy;
5  using org.omg.dds.domain;
6  using org.omg.dds.domain.modifiable;
7  using System;
8  using System.Collections;
9  using System.Collections.Generic;
10 using System.Linq;
11 using System.Text;
12 using System.Threading.Tasks;
13
14 namespace Doopec.Dds.Domain
15 {
16     public class DomainParticipantFactoryQosImpl : EntityQosImpl<DomainParticipantFactoryQos,
       ModifiableDomainParticipantFactoryQos>, DomainParticipantFactoryQos
17     {
18
19         public static  DomainParticipantFactoryQosImpl ConvertTo(Doopec.Configuration.Dds.
       DomainParticipantFactoryQoS qos, Bootstrap boostrap)
20         {
21             DomainParticipantFactoryQosImpl rst = new DomainParticipantFactoryQosImpl(boostrap);
22             rst.EntityFactoryQosPolicy = new EntityFactoryQosPolicyImpl(qos.EntityFactory.
       AutoenableCreatedEntities, boostrap);
23             return rst;
24         }
25
26         public DomainParticipantFactoryQosImpl(Bootstrap boostrap) : base(boostrap)
27         {
28
29         }
30
31         protected internal EntityFactoryQosPolicy EntityFactoryQosPolicy { get; set; }
32
33         public override ModifiableDomainParticipantFactoryQos Modify()
34         {
35             throw new NotImplementedException();
36         }
37
38         public EntityFactoryQosPolicy GetEntityFactory()
39         {
40             return EntityFactoryQosPolicy;
41         }
42     }
43 }
44
```

```
1  using Doopec.Dds.Core;
2  using Doopec.Dds.Pub;
3  using Doopec.Dds.Sub;
4  using Doopec.Dds.Topic;
5  using Doopec.Dds.Utils;
6  using Doopec.Rtps;
7  using Doopec.Rtps.RtpsTransport;
8  using Doopec.Rtps.SharedMem;
9  using Doopec.Rtps.Structure;
10 using Mina.Filter.Statistic;
11 using org.omg.dds.core;
12 using org.omg.dds.core.modifiable;
13 using org.omg.dds.domain;
14 using org.omg.dds.pub;
15 using org.omg.dds.sub;
16 using org.omg.dds.topic;
17 using org.omg.dds.type;
18 using Rtps.Structure;
19 using System;
20 using System.Collections.Generic;
21 using GUID = Rtps.Structure.Types.GUID;
22
23 namespace Doopec.Dds.Domain
24 {
25     public class DomainParticipantImpl : EntityImpl<DomainParticipant,
26                                                     DomainParticipantListener,
27                                                     DomainParticipantQos>, DomainParticipant, IComparable
28     {
29         private int domainId_ = 0;
30         private List<Publisher> publishers_ = new List<Publisher>();
31         private List<Subscriber> subscribers_ = new List<Subscriber>();
32         private List<ITopic> topics_ = new List<ITopic>();
33
34         private ParticipantImpl rtpsParticipant;
35
36         public GUID ParticipantGuid
37         {
38             get
39             {
40                 return rtpsParticipant.Guid;
41             }
42         }
43         public int RtpsParticipantId
44         {
45             get
46             {
47                 return rtpsParticipant.ParticipantId;
48             }
49         }
50
51         public DomainParticipantImpl(Bootstrap bootstrap)
52             : this(0, bootstrap)
53         {
54         }
55
56         public DomainParticipantImpl(int domainId, Bootstrap bootstrap)
57             : this(domainId, null, null, bootstrap)
58         {
59         }
60
61         public DomainParticipantImpl(int domainId, DomainParticipantQos qos, DomainParticipantListener
    listener, Bootstrap bootstrap)
62             : base(bootstrap)
63         {
64             lock (typeof(DomainParticipantImpl))
65             {
66                 int nextId = 0;
67                 DomainParticipantImpl lastParticipant = DiscoveryService.Instance.LookupParticipant
    (domainId) as DomainParticipantImpl;
68                 if (lastParticipant != null)
69                     nextId = lastParticipant.rtpsParticipant.ParticipantId + 1;
70                 rtpsParticipant = new ParticipantImpl(domainId, nextId);
71             }
72
```

```
73              this.domainId_ = domainId;
74              if (qos != null)
75                  this.SetQos(qos);
76              else
77                  // Check default values for qos
78                  throw new NotImplementedException();
79              if (listener != null)
80                  this.SetListener(listener);
81
82              DiscoveryService.Instance.RegisterParticipant(this);
83          }
84          /// <summary>
85          /// Enable the domainParticipant
86          /// </summary>
87          public override void Enable()
88          {
89              base.Enable();
90              rtpsParticipant.Enable();
91          }
92
93          public Publisher CreatePublisher()
94          {
95              Publisher pub = new PublisherImpl(null, null, this, this.Bootstrap);
96              AddPublisher(pub);
97              return pub;
98          }
99
100         public Publisher CreatePublisher(PublisherQos qos, PublisherListener listener, ICollection<Type> ↙
    statuses)
101         {
102             throw new NotImplementedException();
103         }
104
105         public Publisher CreatePublisher(string qosLibraryName, string qosProfileName, PublisherListener ↙
    listener, ICollection<Type> statuses)
106         {
107             throw new NotImplementedException();
108         }
109
110         public Subscriber CreateSubscriber()
111         {
112             Subscriber sub = new SubscriberImpl(null, null, this, this.Bootstrap);
113             AddSubscriber(sub);
114             return sub;
115         }
116
117         public Subscriber CreateSubscriber(SubscriberQos qos, SubscriberListener listener, ICollection ↙
    <Type> statuses)
118         {
119             throw new NotImplementedException();
120         }
121
122         public Subscriber CreateSubscriber(string qosLibraryName, string qosProfileName, ↙
    SubscriberListener listener, ICollection<Type> statuses)
123         {
124             throw new NotImplementedException();
125         }
126
127         public Subscriber BuiltinSubscriber
128         {
129             get
130             {
131                 throw new NotImplementedException();
132             }
133
134         }
135
136         public Topic<TYPE> CreateTopic<TYPE>(string topicName)
137         {
138             //DomainParticipant.TOPIC_QOS_DEFAULT, null,  StatusMask.STATUS_MASK_NONE
139             // look for this topic. Check id it already exists
140             // raise exception if it does exist.
141
142             // Not exists another topic qith the same name
```

```
143             Topic<TYPE> topic = new TopicImpl<TYPE>(topicName, null, null, this);
144             AddTopic(topic);
145             return topic;
146         }
147 
148 
149         public Topic<TYPE> CreateTopic<TYPE>(string topicName, Type type, TopicQos qos, TopicListener
    <TYPE> listener, ICollection<Type> statuses)
150         {
151             throw new NotImplementedException();
152         }
153 
154         public Topic<TYPE> CreateTopic<TYPE>(string topicName, string qosLibraryName, string
    qosProfileName, TopicListener<TYPE> listener, ICollection<Type> statuses)
155         {
156             throw new NotImplementedException();
157         }
158 
159 
160         public Topic<TYPE> CreateTopic<TYPE>(string topicName, TypeSupport<TYPE> type)
161         {
162             throw new NotImplementedException();
163         }
164 
165         public Topic<TYPE> CreateTopic<TYPE>(string topicName, TypeSupport<TYPE> type, TopicQos qos,
    TopicListener<TYPE> listener, ICollection<Type> statuses)
166         {
167             throw new NotImplementedException();
168         }
169 
170         public Topic<TYPE> CreateTopic<TYPE>(string topicName, TypeSupport<TYPE> type, string
    qosLibraryName, string qosProfileName, TopicListener<TYPE> listener, ICollection<Type> statuses)
171         {
172             throw new NotImplementedException();
173         }
174 
175         public Topic<TYPE> FindTopic<TYPE>(string topicName, Duration timeout)
176         {
177             throw new NotImplementedException();
178         }
179 
180         public Topic<TYPE> FindTopic<TYPE>(string topicName, long timeout, TimeUnit unit)
181         {
182             throw new NotImplementedException();
183         }
184 
185         public TopicDescription<TYPE> LookupTopicDescription<TYPE>(string name)
186         {
187             throw new NotImplementedException();
188         }
189 
190         public ContentFilteredTopic<TYPE> CreateContentFilteredTopic<TYPE>(string name, Topic<TYPE>
    relatedTopic, string filterExpression, IList<string> expressionParameters)
191         {
192             throw new NotImplementedException();
193         }
194 
195         public MultiTopic<TYPE> CreateMultiTopic<TYPE>(string name, string typeName, string
    subscriptionExpression, List<string> expressionParameters)
196         {
197             throw new NotImplementedException();
198         }
199 
200         public void CloseContainedEntities()
201         {
202             throw new NotImplementedException();
203         }
204 
205         public void IgnoreParticipant(InstanceHandle handle)
206         {
207             throw new NotImplementedException();
208         }
209 
210         public void IgnoreTopic(InstanceHandle handle)
```

```
211         {
212             throw new NotImplementedException();
213         }
214 
215         public void IgnorePublication(InstanceHandle handle)
216         {
217             throw new NotImplementedException();
218         }
219 
220         public void IgnoreSubscription(InstanceHandle handle)
221         {
222             throw new NotImplementedException();
223         }
224 
225         public int DomainId
226         {
227             get
228             {
229                 return this.domainId_;
230             }
231         }
232 
233         public void AssertLiveliness()
234         {
235             throw new NotImplementedException();
236         }
237 
238         public PublisherQos DefaultPublisherQos
239         {
240             get { throw new NotImplementedException(); }
241             set { throw new NotImplementedException(); }
242         }
243 
244         public void SetDefaultPublisherQos(string qosLibraryName, string qosProfileName)
245         {
246             throw new NotImplementedException();
247         }
248 
249         public SubscriberQos DefaultSubscriberQos
250         {
251             get { throw new NotImplementedException(); }
252             set { throw new NotImplementedException(); }
253         }
254 
255         public void SetDefaultSubscriberQos(string qosLibraryName, string qosProfileName)
256         {
257             throw new NotImplementedException();
258         }
259 
260         public TopicQos DefaultTopicQos
261         {
262             get { throw new NotImplementedException(); }
263             set { throw new NotImplementedException(); }
264         }
265 
266         public void SetDefaultTopicQos(string qosLibraryName, string qosProfileName)
267         {
268             throw new NotImplementedException();
269         }
270 
271         public ICollection<InstanceHandle> GetDiscoveredParticipants(ICollection<InstanceHandle>        ↙
    participantHandles)
272         {
273             throw new NotImplementedException();
274         }
275 
276         public ParticipantBuiltinTopicData GetDiscoveredParticipantData(ParticipantBuiltinTopicData      ↙
    participantData, InstanceHandle participantHandle)
277         {
278             throw new NotImplementedException();
279         }
280 
281         public ICollection<InstanceHandle> GetDiscoveredTopics(ICollection<InstanceHandle> topicHandles)
282         {
```

```
283             throw new NotImplementedException();
284         }
285 
286         public TopicBuiltinTopicData GetDiscoveredTopicData(TopicBuiltinTopicData topicData,
    InstanceHandle topicHandle)
287         {
288             throw new NotImplementedException();
289         }
290 
291         public bool ContainsEntity(InstanceHandle handle)
292         {
293             throw new NotImplementedException();
294         }
295 
296         public ModifiableTime CurrentTime(ModifiableTime currentTime)
297         {
298             throw new NotImplementedException();
299         }
300         public override void Close()
301         {
302             rtpsParticipant.Close();
303             DiscoveryService.Instance.UnregisterParticipant(this);
304             base.Close();
305         }
306 
307         public override void Retain()
308         {
309             throw new NotImplementedException();
310         }
311 
312 
313         #region API Extensions
314 
315         internal protected void AddPublisher(Publisher pub)
316         {
317             this.publishers_.Add(pub);
318         }
319 
320         internal protected void DeletePublisher(Publisher pub)
321         {
322             this.publishers_.Remove(pub);
323         }
324 
325         internal protected void AddSubscriber(Subscriber sub)
326         {
327             this.subscribers_.Add(sub);
328         }
329 
330         internal protected void DeleteSubscriber(Subscriber sub)
331         {
332             this.subscribers_.Remove(sub);
333         }
334 
335         internal protected void AddTopic(ITopic topic)
336         {
337             this.topics_.Add(topic);
338         }
339 
340         internal protected void DeleteTopic(ITopic topic)
341         {
342             this.topics_.Remove(topic);
343         }
344         #endregion
345 
346         public Topic<TYPE> FindTopic<TYPE>(string topicName, long timeout, global::DDS.ConversionUtils.
    TimeUnit unit)
347         {
348             throw new NotImplementedException();
349         }
350 
351         public override string ToString()
352         {
353             return String.Format("Participant ID:{0} at Domain {1}", this.rtpsParticipant.Guid, this.
    domainId_);
```

```
354         }
355 
356         public override bool Equals(object obj)
357         {
358             if (obj != null && obj is DomainParticipantImpl)
359             {
360                 DomainParticipantImpl other = obj as DomainParticipantImpl;
361                 if (this.rtpsParticipant != null != null && other.rtpsParticipant != null &&
362                     this.rtpsParticipant.Guid == other.rtpsParticipant.Guid)
363                 {
364                     return true;
365                 }
366             }
367 
368             return false;
369         }
370         public override int GetHashCode()
371         {
372             return this.rtpsParticipant.Guid.GetHashCode();
373         }
374 
375         /// <summary>
376         /// Compares the current instance with another object of the same type and returns
377         /// an integer that indicates whether the current instance precedes, follows,
378         ///  or occurs in the same position in the sort order as the other object.
379         /// </summary>
380         /// <param name="obj">An object to compare with this instance.</param>
381         /// <returns>
382         /// A value that indicates the relative order of the objects being compared.
383         /// The return value has these meanings: Value Meaning Less than zero This instance
384         /// precedes obj in the sort order. Zero This instance occurs in the same position
385         /// in the sort order as obj. Greater than zero This instance follows obj in
386         /// the sort order.
387         ///</returns>
388         public int CompareTo(object obj)
389         {
390             if (obj == null || !(obj is DomainParticipantImpl))
391                 throw new System.ArgumentException();
392 
393             DomainParticipantImpl other = (DomainParticipantImpl)obj;
394             if (this.rtpsParticipant != null != null && other.rtpsParticipant != null)
395             {
396                 return this.rtpsParticipant.ParticipantId - other.rtpsParticipant.ParticipantId;
397             }
398             else
399                 throw new System.ArgumentException();
400         }
401     }
402 }
403 
```

```
1  using Doopec.Dds.Domain;
2  using Doopec.Dds.Domain.modifiable;
3  using Doopec.DDS.Core;
4  using Doopec.DDS.Core.Policy;
5  using org.omg.dds.core;
6  using org.omg.dds.core.policy;
7  using org.omg.dds.domain;
8  using org.omg.dds.domain.modifiable;
9  using System;
10 using System.Text;
11
12 namespace Doopec.DDS.Domain
13 {
14     public class DomainParticipantQosImpl : EntityQosImpl<DomainParticipantQos,
       ModifiableDomainParticipantQos>, DomainParticipantQos
15     {
16         public readonly DomainParticipantQosImpl DDS_PARTICIPANT_QOS_DEFAULT;
17
18         public UserDataQosPolicy UserData {get; internal set;}
19         public EntityFactoryQosPolicy EntityFactoryQosPolicy { get; internal set; }
20
21         public DomainParticipantQosImpl(Bootstrap boostrap)
22             : base(boostrap)
23         {
24             // TODO
25             // DDS_PARTICIPANT_QOS_DEFAULT = new DomainParticipantQosImpl(boostrap);
26             UserData = new UserDataQosPolicyImpl(boostrap);
27             EntityFactoryQosPolicy = new EntityFactoryQosPolicyImpl(boostrap);
28         }
29
30         public DomainParticipantQosImpl(UserDataQosPolicy userData, EntityFactoryQosPolicy
       entityFactoryQosPolicy, Bootstrap boostrap)
31             : base(boostrap)
32         {
33             this.UserData = userData;
34             this.EntityFactoryQosPolicy = entityFactoryQosPolicy;
35         }
36
37         public UserDataQosPolicy GetUserData()
38         {
39             return UserData;
40         }
41
42         public EntityFactoryQosPolicy GetEntityFactory()
43         {
44             return EntityFactoryQosPolicy;
45         }
46
47         public override ModifiableDomainParticipantQos Modify()
48         {
49             return new ModifiableDomainParticipantQosImpl(this);
50             //throw new NotImplementedException();
51         }
52
53         public static DomainParticipantQosImpl ConvertTo(Doopec.Configuration.Dds.DomainParticipantQoS
       qosConfig, Bootstrap boostrap)
54         {
55             DomainParticipantQosImpl rst = new DomainParticipantQosImpl(boostrap);
56             rst.UserData = new UserDataQosPolicyImpl(UTF8Encoding.Unicode.GetBytes(qosConfig.UserData.
       Value), boostrap);
57             rst.EntityFactoryQosPolicy = new EntityFactoryQosPolicyImpl(qosConfig.EntityFactory.
       AutoenableCreatedEntities, boostrap);
58             return rst;
59         }
60     }
61 }
62
```

```
1  using org.omg.dds.core;
2  using org.omg.dds.core.policy;
3  using org.omg.dds.pub;
4  using org.omg.dds.pub.modifiable;
5  using org.omg.dds.topic;
6  using System;
7  using System.Collections;
8  using System.Collections.Generic;
9  using System.Linq;
10 using System.Text;
11 using System.Threading.Tasks;
12
13 namespace Doopec.Dds.Pub.modifiable
14 {
15     class ModifiableDataWriterQosImpl : ModifiableDataWriterQos
16     {
17         private DataWriterQosImpl innerQos;
18
19         public ModifiableDataWriterQosImpl(DataWriterQosImpl qos)
20          {
21             this.innerQos = qos;
22          }
23
24         public ModifiableDataWriterQosImpl(Bootstrap boostrap)
25
26         {
27             this.innerQos = new DataWriterQosImpl(boostrap);
28         }
29
30         public ModifiableDataWriterQos SetDurability(DurabilityQosPolicy durability)
31         {
32             this.innerQos.Durability = durability;
33             return this;
34         }
35
36         public ModifiableDataWriterQos SetDurabilityService(DurabilityServiceQosPolicy durabilityService)
37         {
38             this.innerQos.DurabilityService = durabilityService;
39             return this;
40         }
41
42         public ModifiableDataWriterQos SetDeadline(DeadlineQosPolicy deadline)
43         {
44             this.innerQos.Deadline = deadline;
45             return this;
46         }
47
48         public ModifiableDataWriterQos SetLatencyBudget(LatencyBudgetQosPolicy latencyBudget)
49         {
50             this.innerQos.LatencyBudget = latencyBudget;
51             return this;
52         }
53
54         public ModifiableDataWriterQos SetLiveliness(LivelinessQosPolicy liveliness)
55         {
56             this.innerQos.Liveliness = liveliness;
57             return this;
58         }
59
60         public ModifiableDataWriterQos SetReliability(ReliabilityQosPolicy reliability)
61         {
62             this.innerQos.Reliability = reliability;
63             return this;
64         }
65
66         public ModifiableDataWriterQos SetDestinationOrder(DestinationOrderQosPolicy destinationOrder)
67         {
68             this.innerQos.DestinationOrder = destinationOrder;
69             return this;
70         }
71
72         public ModifiableDataWriterQos SetHistory(HistoryQosPolicy history)
73         {
74             this.innerQos.History = history;
```

```
75              return this;
76          }
77
78          public ModifiableDataWriterQos SetResourceLimits(ResourceLimitsQosPolicy resourceLimits)
79          {
80              this.innerQos.ResourceLimits = resourceLimits;
81              return this;
82          }
83
84          public ModifiableDataWriterQos SetTransportPriority(TransportPriorityQosPolicy transportPriority)
85          {
86              this.innerQos.TransportPriority = transportPriority;
87              return this;
88          }
89
90          public ModifiableDataWriterQos SetLifespan(LifespanQosPolicy lifespan)
91          {
92              this.innerQos.Lifespan = lifespan;
93              return this;
94          }
95
96          public ModifiableDataWriterQos SetUserData(UserDataQosPolicy userData)
97          {
98              this.innerQos.UserData = userData;
99              return this;
100         }
101
102         public ModifiableDataWriterQos SetOwnership(OwnershipQosPolicy ownership)
103         {
104             this.innerQos.Ownership = ownership;
105             return this;
106         }
107
108         public ModifiableDataWriterQos SetOwnershipStrength(OwnershipStrengthQosPolicy ownershipStrength)
109         {
110             this.innerQos.OwnershipStrength = ownershipStrength;
111             return this;
112         }
113
114         public ModifiableDataWriterQos SetWriterDataLifecycle(WriterDataLifecycleQosPolicy writerDataLifecycle)
115         {
116             this.innerQos.WriterDataLifecycle = writerDataLifecycle;
117             return this;
118         }
119
120         public ModifiableDataWriterQos SetRepresentation(DataRepresentationQosPolicy representation)
121         {
122             this.innerQos.Representation = representation;
123             return this;
124         }
125
126         public ModifiableDataWriterQos SetTypeConsistency(TypeConsistencyEnforcementQosPolicy typeConsistency)
127         {
128             this.innerQos.TypeConsistency = typeConsistency;
129             return this;
130         }
131
132         public ModifiableDataWriterQos CopyFrom(TopicQos src)
133         {
134             throw new NotImplementedException();
135         }
136
137         public POLICY put<POLICY>(QosPolicyId key, POLICY value) where POLICY : QosPolicy
138         {
139             throw new NotImplementedException();
140         }
141
142         public ModifiableDataWriterQos CopyFrom(DataWriterQos other)
143         {
144             throw new NotImplementedException();
145         }
146
```

```
147         public DataWriterQos FinishModification()
148         {
149             return this.innerQos;
150         }
151 
152         public DurabilityQosPolicy GetDurability()
153         {
154             return this.innerQos.GetDurability();
155         }
156 
157         public DurabilityServiceQosPolicy GetDurabilityService()
158         {
159             return this.innerQos.GetDurabilityService();
160         }
161 
162         public DeadlineQosPolicy GetDeadline()
163         {
164             return this.innerQos.GetDeadline();
165         }
166 
167         public LatencyBudgetQosPolicy GetLatencyBudget()
168         {
169             return this.innerQos.GetLatencyBudget();
170         }
171 
172         public LivelinessQosPolicy GetLiveliness()
173         {
174             return this.innerQos.GetLiveliness();
175         }
176 
177         public ReliabilityQosPolicy GetReliability()
178         {
179             return this.innerQos.GetReliability();
180         }
181 
182         public DestinationOrderQosPolicy GetDestinationOrder()
183         {
184             return this.innerQos.GetDestinationOrder();
185         }
186 
187         public HistoryQosPolicy GetHistory()
188         {
189             return this.innerQos.GetHistory();
190         }
191 
192         public ResourceLimitsQosPolicy GetResourceLimits()
193         {
194             return this.innerQos.GetResourceLimits();
195         }
196 
197         public TransportPriorityQosPolicy GetTransportPriority()
198         {
199             return this.innerQos.GetTransportPriority();
200         }
201 
202         public LifespanQosPolicy GetLifespan()
203         {
204             return this.innerQos.GetLifespan();
205         }
206 
207         public UserDataQosPolicy GetUserData()
208         {
209             return this.innerQos.GetUserData();
210         }
211 
212         public OwnershipQosPolicy GetOwnership()
213         {
214             return this.innerQos.GetOwnership();
215         }
216 
217         public OwnershipStrengthQosPolicy GetOwnershipStrength()
218         {
219             return this.innerQos.GetOwnershipStrength();
220         }
```

```
221
222         public WriterDataLifecycleQosPolicy GetWriterDataLifecycle()
223         {
224             return this.innerQos.GetWriterDataLifecycle();
225         }
226
227         public DataRepresentationQosPolicy GetRepresentation()
228         {
229             return this.innerQos.GetRepresentation();
230         }
231
232         public TypeConsistencyEnforcementQosPolicy GetTypeConsistency()
233         {
234             return this.innerQos.GetTypeConsistency();
235         }
236
237         public POLICY Get<POLICY>(QosPolicyId id) where POLICY : QosPolicy
238         {
239             throw new NotImplementedException();
240         }
241
242         public QosPolicy Put(QosPolicyId key, QosPolicy value)
243         {
244             throw new NotImplementedException();
245         }
246
247         public QosPolicy Remove(object key)
248         {
249             throw new NotImplementedException();
250         }
251
252         public void Clear()
253         {
254             this.innerQos.Clear();
255         }
256
257         public Bootstrap GetBootstrap()
258         {
259             return this.innerQos.GetBootstrap();
260         }
261
262         public void Add(QosPolicyId key, QosPolicy value)
263         {
264             throw new NotImplementedException();
265         }
266
267         public bool ContainsKey(QosPolicyId key)
268         {
269             throw new NotImplementedException();
270         }
271
272         public ICollection<QosPolicyId> Keys
273         {
274             get { throw new NotImplementedException(); }
275         }
276
277         public bool Remove(QosPolicyId key)
278         {
279             throw new NotImplementedException();
280         }
281
282         public bool TryGetValue(QosPolicyId key, out QosPolicy value)
283         {
284             throw new NotImplementedException();
285         }
286
287         public ICollection<QosPolicy> Values
288         {
289             get { throw new NotImplementedException(); }
290         }
291
292         public QosPolicy this[QosPolicyId key]
293         {
294             get
```

```
295             {
296                 return this.innerQos[key];
297             }
298             set
299             {
300                 this.innerQos[key] = value;
301             }
302         }
303 
304         public void Add(KeyValuePair<QosPolicyId, QosPolicy> item)
305         {
306             this.innerQos.Add(item);
307         }
308 
309         public bool Contains(KeyValuePair<QosPolicyId, QosPolicy> item)
310         {
311             return this.innerQos.Contains(item);
312         }
313 
314         public void CopyTo(KeyValuePair<QosPolicyId, QosPolicy>[] array, int arrayIndex)
315         {
316             throw new NotImplementedException();
317         }
318 
319         public int Count
320         {
321             get { return this.innerQos.Count; }
322         }
323 
324         public bool IsReadOnly
325         {
326             get { return this.innerQos.IsReadOnly; }
327         }
328 
329         public bool Remove(KeyValuePair<QosPolicyId, QosPolicy> item)
330         {
331             return this.innerQos.Remove(item);
332         }
333 
334         public IEnumerator<KeyValuePair<QosPolicyId, QosPolicy>> GetEnumerator()
335         {
336             throw new NotImplementedException();
337         }
338 
339         IEnumerator IEnumerable.GetEnumerator()
340         {
341             return this.innerQos.GetEnumerator();
342         }
343 
344         public ModifiableDataWriterQos Modify()
345         {
346             return this;
347         }
348     }
349 }
```

```
1  using DDS.ConversionUtils;
2  using Doopec.Dds.Domain;
3  using Doopec.Rtps.Behavior;
4  using Doopec.Rtps.Structure;
5  using org.omg.dds.core;
6  using org.omg.dds.pub;
7  using org.omg.dds.topic;
8  using Rtps.Messages;
9  using Rtps.Structure;
10 using Rtps.Structure.Types;
11 using System;
12 using System.Collections.Generic;
13 using InstanceHandle = Rtps.Structure.Types.InstanceHandle;
14
15 namespace Doopec.Dds.Pub
16 {
17     public class DataWriterImpl<TYPE> : DataWriter<TYPE>
18     {
19         Topic<TYPE> topic_;
20         Publisher pub_;
21         DataWriterListener<TYPE> listener;
22         DataWriterQos qos;
23
24         /// <summary>
25         ///
26         /// </summary>
27         /// <typeparam name="T"></typeparam>
28         /// <param name="?"></param>
29         /// <returns></returns>
30         protected readonly Writer<TYPE> rtpsWriter;
31
32         /// <summary>
33         ///
34         /// </summary>
35         /// <typeparam name="T"></typeparam>
36         /// <param name="?"></param>
37         /// <returns></returns>
38         protected readonly HistoryCache<TYPE> historyCache;
39
40         public DataWriterImpl(Publisher pub, Topic<TYPE> topic)
41             : this(pub, topic, pub.GetDefaultDataWriterQos(), null, null)
42         {
43         }
44
45         public DataWriterImpl(Publisher pub, Topic<TYPE> topic, DataWriterQos qos, DataWriterListener
       <TYPE> listener, ICollection<Type> statuses)
46         {
47             this.pub_ = pub;
48             this.topic_ = topic;
49             this.listener = listener;
50
51             this.rtpsWriter = new RtpsStatefulWriter<TYPE>((pub.GetParent() as DomainParticipantImpl).
       ParticipantGuid);
52         }
53
54
55         public Type GetType()
56         {
57             return topic_.GetType();
58         }
59
60         public DataWriter<OTHER> Cast<OTHER>()
61         {
62             throw new NotImplementedException();
63         }
64
65         public org.omg.dds.topic.Topic<TYPE> GetTopic()
66         {
67             return topic_;
68         }
69
70         public void WaitForAcknowledgments(org.omg.dds.core.Duration maxWait)
71         {
72             throw new NotImplementedException();
```

```
73          }
74
75          public void WaitForAcknowledgments(long maxWait, TimeUnit unit)
76          {
77              throw new NotImplementedException();
78          }
79
80          public org.omg.dds.core.status.LivelinessLostStatus<TYPE> GetLivelinessLostStatus(org.omg.dds.
    core.status.LivelinessLostStatus<TYPE> status)
81          {
82              throw new NotImplementedException();
83          }
84
85          public org.omg.dds.core.status.OfferedDeadlineMissedStatus<TYPE> GetOfferedDeadlineMissedStatus
    (org.omg.dds.core.status.OfferedDeadlineMissedStatus<TYPE> status)
86          {
87              throw new NotImplementedException();
88          }
89
90          public org.omg.dds.core.status.OfferedIncompatibleQosStatus<TYPE> GetOfferedIncompatibleQosStatus
    (org.omg.dds.core.status.OfferedIncompatibleQosStatus<TYPE> status)
91          {
92              throw new NotImplementedException();
93          }
94
95          public org.omg.dds.core.status.PublicationMatchedStatus<TYPE> GetPublicationMatchedStatus(org.omg
    .dds.core.status.PublicationMatchedStatus<TYPE> status)
96          {
97              throw new NotImplementedException();
98          }
99
100         public void AssertLiveliness()
101         {
102             throw new NotImplementedException();
103         }
104
105         public ICollection<org.omg.dds.core.InstanceHandle> GetMatchedSubscriptions(ICollection<org.omg.
    dds.core.InstanceHandle> subscriptionHandles)
106         {
107             throw new NotImplementedException();
108         }
109
110         public org.omg.dds.topic.SubscriptionBuiltinTopicData GetMatchedSubscriptionData(org.omg.dds.
    topic.SubscriptionBuiltinTopicData subscriptionData, org.omg.dds.core.InstanceHandle
    subscriptionHandle)
111         {
112             throw new NotImplementedException();
113         }
114
115         public org.omg.dds.core.InstanceHandle RegisterInstance(TYPE instanceData)
116         {
117             throw new NotImplementedException();
118         }
119
120         public org.omg.dds.core.InstanceHandle RegisterInstance(TYPE instanceData, org.omg.dds.core.Time
    sourceTimestamp)
121         {
122             throw new NotImplementedException();
123         }
124
125         public org.omg.dds.core.InstanceHandle RegisterInstance(TYPE instanceData, long sourceTimestamp,
    TimeUnit unit)
126         {
127             throw new NotImplementedException();
128         }
129
130         public void UnregisterInstance(org.omg.dds.core.InstanceHandle handle)
131         {
132             throw new NotImplementedException();
133         }
134
135         public void UnregisterInstance(org.omg.dds.core.InstanceHandle handle, TYPE instanceData)
136         {
137             throw new NotImplementedException();
```

```
138         }
139
140         public void UnregisterInstance(org.omg.dds.core.InstanceHandle handle, TYPE instanceData, org.omg
    .dds.core.Time sourceTimestamp)
141         {
142             throw new NotImplementedException();
143         }
144
145         public void UnregisterInstance(org.omg.dds.core.InstanceHandle handle, TYPE instanceData, long
    sourceTimestamp, TimeUnit unit)
146         {
147             throw new NotImplementedException();
148         }
149
150         public void Write(TYPE instanceData)
151         {
152             // A single tick represents one hundred nanoseconds
153             long ts = System.DateTime.Now.Ticks / (TimeSpan.TicksPerMillisecond / 1000);
154             this.Write(instanceData, ts, TimeUnit.NANOSECONDS);
155         }
156
157         public void Write(TYPE instanceData, Time sourceTimestamp)
158         {
159             throw new NotImplementedException();
160         }
161
162         public void Write(TYPE instanceData, long sourceTimestamp, TimeUnit unit)
163         {
164             // TODO. Implements timestamp and timeunit in Write
165             CacheChange<TYPE> change = rtpsWriter.NewChange(ChangeKind.ALIVE, new Data(instanceData), new
     InstanceHandle());
166             rtpsWriter.HistoryCache.AddChange(change);
167         }
168
169         public void Write(TYPE instanceData, org.omg.dds.core.InstanceHandle handle)
170         {
171             throw new NotImplementedException();
172         }
173
174         public void Write(TYPE instanceData, org.omg.dds.core.InstanceHandle handle, org.omg.dds.core.
    Time sourceTimestamp)
175         {
176             throw new NotImplementedException();
177         }
178
179         public void Write(TYPE instanceData, org.omg.dds.core.InstanceHandle handle, long sourceTimestamp
    , TimeUnit unit)
180         {
181             throw new NotImplementedException();
182         }
183
184         public void Dispose(org.omg.dds.core.InstanceHandle instanceHandle)
185         {
186             throw new NotImplementedException();
187         }
188
189         public void Dispose(org.omg.dds.core.InstanceHandle instanceHandle, TYPE instanceData)
190         {
191             throw new NotImplementedException();
192         }
193
194         public void Dispose(org.omg.dds.core.InstanceHandle instanceHandle, TYPE instanceData, org.omg.
    dds.core.Time sourceTimestamp)
195         {
196             throw new NotImplementedException();
197         }
198
199         public void Dispose(org.omg.dds.core.InstanceHandle instanceHandle, TYPE instanceData, long
    sourceTimestamp, TimeUnit unit)
200         {
201             throw new NotImplementedException();
202         }
203
204         public TYPE GetKeyValue(TYPE keyHolder, org.omg.dds.core.InstanceHandle handle)
```

```
205         {
206             throw new NotImplementedException();
207         }
208 
209         public org.omg.dds.core.modifiable.ModifiableInstanceHandle LookupInstance(org.omg.dds.core.
    modifiable.ModifiableInstanceHandle handle, TYPE keyHolder)
210         {
211             throw new NotImplementedException();
212         }
213 
214         public Publisher GetParent()
215         {
216             return this.pub_;
217         }
218 
219         public DataWriterListener<TYPE> GetListener()
220         {
221             return this.listener;
222         }
223 
224         public void SetListener(DataWriterListener<TYPE> listener)
225         {
226             this.listener = listener;
227         }
228 
229         public DataWriterQos GetQos()
230         {
231             return this.qos;
232         }
233 
234         public void SetQos(DataWriterQos qos)
235         {
236             this.qos = qos;
237         }
238 
239         public void SetQos(string qosLibraryName, string qosProfileName)
240         {
241             throw new NotImplementedException();
242         }
243 
244         public void Enable()
245         {
246             throw new NotImplementedException();
247         }
248 
249         public org.omg.dds.core.StatusCondition<DataWriter<TYPE>> GetStatusCondition()
250         {
251             throw new NotImplementedException();
252         }
253 
254         public ICollection<TYPE> GetStatusChanges<TYPE>(ICollection<TYPE> statuses)
255         {
256             throw new NotImplementedException();
257         }
258 
259         public org.omg.dds.core.InstanceHandle GetInstanceHandle()
260         {
261             throw new NotImplementedException();
262         }
263 
264         public void Close()
265         {
266             throw new NotImplementedException();
267         }
268 
269         public void Retain()
270         {
271             throw new NotImplementedException();
272         }
273 
274         public org.omg.dds.core.Bootstrap GetBootstrap()
275         {
276             throw new NotImplementedException();
277         }
```

```
278
279
280     }
281 }
282
```

```
using Doopec.Dds.Pub.modifiable;
using Doopec.DDS.Core;
using org.omg.dds.core;
using org.omg.dds.core.policy;
using org.omg.dds.pub;
using org.omg.dds.pub.modifiable;
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Doopec.Dds.Pub
{
    public class DataWriterQosImpl : EntityQosImpl<DataWriterQos, ModifiableDataWriterQos>, DataWriterQos
    {
        public DataWriterQosImpl(Bootstrap boostrap)
            : base(boostrap)
        {
        }

        public DataWriterQosImpl(DurabilityQosPolicy durability, Bootstrap boostrap)
            : base(boostrap)
        {
            this.Durability = durability;
        }
        public DataWriterQosImpl(DurabilityServiceQosPolicy durabilityService, Bootstrap boostrap)
            : base(boostrap)
        {
            this.DurabilityService = durabilityService;
        }
        public DataWriterQosImpl(ReliabilityQosPolicy reliability, Bootstrap boostrap)
            : base(boostrap)
        {
            this.Reliability = reliability;
        }
        public DataWriterQosImpl(DestinationOrderQosPolicy destinationOrder, Bootstrap boostrap)
            : base(boostrap)
        {
            this.DestinationOrder = destinationOrder;
        }


        public DurabilityQosPolicy GetDurability()
        {
            return Durability ;
        }

        public DurabilityServiceQosPolicy GetDurabilityService()
        {
            return DurabilityService ;
        }

        public DeadlineQosPolicy GetDeadline()
        {
            return Deadline ;
        }

        public LatencyBudgetQosPolicy GetLatencyBudget()
        {
            return LatencyBudget ;
        }

        public LivelinessQosPolicy GetLiveliness()
        {
            return Liveliness;
        }

        public ReliabilityQosPolicy GetReliability()
        {
            return Reliability;
        }

```

```
75          public DestinationOrderQosPolicy GetDestinationOrder()
76          {
77              return DestinationOrder ;
78          }
79
80          public HistoryQosPolicy GetHistory()
81          {
82              return History;
83          }
84
85          public ResourceLimitsQosPolicy GetResourceLimits()
86          {
87              return ResourceLimits ;
88          }
89
90          public TransportPriorityQosPolicy GetTransportPriority()
91          {
92              return TransportPriority ;
93          }
94
95          public LifespanQosPolicy GetLifespan()
96          {
97              return Lifespan ;
98          }
99
100         public UserDataQosPolicy GetUserData()
101         {
102             return UserData ;
103         }
104
105         public OwnershipQosPolicy GetOwnership()
106         {
107             return Ownership ;
108         }
109
110         public OwnershipStrengthQosPolicy GetOwnershipStrength()
111         {
112             return OwnershipStrength ;
113         }
114
115         public WriterDataLifecycleQosPolicy GetWriterDataLifecycle()
116         {
117             return WriterDataLifecycle;
118         }
119
120         public DataRepresentationQosPolicy GetRepresentation()
121         {
122             return Representation;
123         }
124
125         public TypeConsistencyEnforcementQosPolicy GetTypeConsistency()
126         {
127             return TypeConsistency;
128         }
129
130         public new IEnumerator GetEnumerator()
131         {
132             throw new NotImplementedException();
133         }
134
135         public override ModifiableDataWriterQos Modify()
136         {
137           // throw new NotImplementedException();
138             return new  ModifiableDataWriterQosImpl(this);
139         }
140
141         internal DurabilityQosPolicy Durability { get; set; }
142         internal DurabilityServiceQosPolicy DurabilityService { get; set; }
143         internal DeadlineQosPolicy Deadline { get; set; }
144         internal LatencyBudgetQosPolicy LatencyBudget { get; set; }
145         internal DestinationOrderQosPolicy DestinationOrder { get; set; }
146         internal HistoryQosPolicy History { get; set; }
147         internal ResourceLimitsQosPolicy ResourceLimits { get; set; }
148         internal TransportPriorityQosPolicy TransportPriority { get; set; }
```

```
149         internal LifespanQosPolicy Lifespan { get; set; }
150         internal UserDataQosPolicy UserData { get; set; }
151         internal OwnershipQosPolicy Ownership { get; set; }
152         internal OwnershipStrengthQosPolicy OwnershipStrength { get; set; }
153         internal WriterDataLifecycleQosPolicy WriterDataLifecycle { get; set; }
154         internal DataRepresentationQosPolicy Representation { get; set; }
155         internal TypeConsistencyEnforcementQosPolicy TypeConsistency { get; set; }
156         internal ReliabilityQosPolicy Reliability { get; set; }
157         internal LivelinessQosPolicy Liveliness { get; set; }
158
159
160     }
161 }
162
```

```
1  using DDS.ConversionUtils;
2  using Doopec.Configuration;
3  using Doopec.Dds.Core;
4  using Doopec.Dds.Core.Policy;
5  using Doopec.DDS.Core.Policy;
6  using org.omg.dds.core;
7  using org.omg.dds.core.policy;
8  using org.omg.dds.core.policy.modifiable;
9  using org.omg.dds.domain;
10 using org.omg.dds.pub;
11 using org.omg.dds.pub.modifiable;
12 using org.omg.dds.topic;
13 using System;
14 using System.Collections;
15 using System.Collections.Generic;
16 using System.Configuration;
17
18 namespace Doopec.Dds.Pub
19 {
20     public class PublisherImpl : Publisher
21     {
22         private PublisherQos qos;
23         private DomainParticipant parent;
24         private DataWriterQosImpl dataWriterqos;
25         private PublisherListener listener;
26         private IList datawriters;
27         private DDSConfigurationSection ddsConfig = Doopec.Configuration.DDSConfigurationSection.Instance;
28         public Bootstrap Bootstrap { get; internal set; }
29
30         public PublisherImpl(PublisherQos qos, PublisherListener listener, DomainParticipant dp, Bootstrap bootstrap)
31         {
32             this.qos = qos;
33             this.listener = listener;
34             this.parent = dp;
35             this.Bootstrap = bootstrap;
36             datawriters = new System.Collections.ArrayList();
37             dataWriterqos = new DataWriterQosImpl(this.GetBootstrap());
38
39             string qosConfigProfile = ddsConfig.Domains[dp.DomainId].QoSProfile.Name;
40             Doopec.Configuration.Dds.QoSProfilePolicy qosProfile = ddsConfig.QoSProfiles[qosConfigProfile];
41
42             if (qosProfile != null)
43             {
44                 Doopec.Configuration.Dds.DataWriterQoS dataWriterProfileQos = qosProfile.DataWriterQoS;
45                 // TODO Assign values to dataWriterqos from configuration
46                 if (dataWriterProfileQos == null)
47                     return;
48                 if (dataWriterProfileQos.Reliability != null)
49                 {
50                     ReliabilityQosPolicyImpl dpqMod = new ReliabilityQosPolicyImpl(this.GetBootstrap());
51                     if (dataWriterProfileQos.Reliability.Kind == Doopec.Configuration.Reliability.RELIABLE)
52                     {
53                         dpqMod = new ReliabilityQosPolicyImpl(ReliabilityQosPolicyKind.RELIABLE, this.GetBootstrap());
54
55                     }
56
57                     else if (dataWriterProfileQos.Reliability.Kind == Doopec.Configuration.Reliability.BEST_EFFORT)
58                         dpqMod = new ReliabilityQosPolicyImpl(ReliabilityQosPolicyKind.BEST_EFFORT, this.GetBootstrap());
59                     dpqMod.MaxBlockingTimeQos = new DurationImpl(this.GetBootstrap(), dataWriterProfileQos.Reliability.MaxBlockingTime);
60                     dataWriterqos.Reliability = dpqMod;
61                 }
62
63                 if (dataWriterProfileQos.Durability != null)
64                 {
65                     DurabilityQosPolicyImpl dpqMod = new DurabilityQosPolicyImpl(this.GetBootstrap());
66                     switch (dataWriterProfileQos.Durability.Kind)
```

```
67                      {
68                          case Durability.PERSISTENT:
69                              dpqMod = new DurabilityQosPolicyImpl(DurabilityQosPolicyKind.PERSISTENT, this↙
      .GetBootstrap());
70                              break;
71                          case Durability.TRANSIENT:
72                              dpqMod = new DurabilityQosPolicyImpl(DurabilityQosPolicyKind.TRANSIENT, this.↙
      GetBootstrap());
73                              break;
74                          case Durability.TRANSIENT_LOCAL:
75                              dpqMod = new DurabilityQosPolicyImpl(DurabilityQosPolicyKind.TRANSIENT_LOCAL,↙
       this.GetBootstrap());
76                              break;
77                          case Durability.VOLATILE:
78                              dpqMod = new DurabilityQosPolicyImpl(DurabilityQosPolicyKind.VOLATILE, this. ↙
      GetBootstrap());
79                              break;
80                      }
81
82                      dataWriterqos.Durability = dpqMod;
83                  }
84                  if (dataWriterProfileQos.Deadline != null)
85                  {
86                      // TODO dataWriterProfileQos.Deadline
87                      DeadlineQosPolicyImpl dpqMod = new DeadlineQosPolicyImpl(this.GetBootstrap());
88
89
90
91                  }
92                  if (dataWriterProfileQos.DurabilityService != null)
93                  {
94                      // TODO dataWriterProfileQos.Deadline
95                      DurabilityServiceQosPolicyImpl dpqMod = new DurabilityServiceQosPolicyImpl(this.     ↙
      GetBootstrap());
96                      switch (dataWriterProfileQos.DurabilityService.HistoryKind)
97                      {
98                          case History.KEEP_ALL:
99                              dpqMod.HistoryQosPolicyKind = HistoryQosPolicyKind.KEEP_ALL;
100                             break;
101                         case History.KEEP_LAST:
102                             dpqMod.HistoryQosPolicyKind = HistoryQosPolicyKind.KEEP_LAST;
103                             break;
104                     }
105
106                 }
107                 if (dataWriterProfileQos.LatencyBudget != null)
108                 {
109                     // TODO dataWriterProfileQos.LatencyBudget
110                     LatencyBudgetQosPolicyImpl dpqMod = new LatencyBudgetQosPolicyImpl(this.GetBootstrap ↙
      ());
111
112
113                 }
114                 if (dataWriterProfileQos.Liveliness != null)
115                 {
116                     // TODO dataWriterProfileQos.Liveliness
117                     LivelinessQosPolicyImpl dpqMod = new LivelinessQosPolicyImpl(this.GetBootstrap());
118                     switch (dataWriterProfileQos.Liveliness.Kind)
119                     {
120                         case Liveliness.AUTOMATIC:
121                             dpqMod = new LivelinessQosPolicyImpl( LivelinessQosPolicyKind.AUTOMATIC,this.↙
      GetBootstrap());
122                             break;
123                         case Liveliness.MANUAL_BY_PARTICIPANT:
124                             dpqMod =new LivelinessQosPolicyImpl( LivelinessQosPolicyKind.                ↙
      MANUAL_BY_PARTICIPANT,this.GetBootstrap());
125                             break;
126                         case Liveliness.MANUAL_BY_TOPIC:
127                             dpqMod = new LivelinessQosPolicyImpl(LivelinessQosPolicyKind.MANUAL_BY_TOPIC,↙
      this.GetBootstrap());
128                             break;
129
130                     }
131                     dpqMod.LeaseDurationQos = new DurationImpl(this.GetBootstrap(), dataWriterProfileQos.↙
```

```
    Liveliness.LeaseDuration);
132                 dataWriterqos.Liveliness = dpqMod;
133
134
135             }
136             if (dataWriterProfileQos.DestinationOrder != null)
137             {
138                 // TODO dataWriterProfileQos.DestinationOrder
139                 DestinationOrderQosPolicyImpl dpqMod = new DestinationOrderQosPolicyImpl(this.       ↙
    GetBootstrap());
140                 switch (dataWriterProfileQos.DestinationOrder.Kind)
141                 {
142                     case DestinationOrder.BY_RECEPTION_TIMESTAMP:
143                         dpqMod =new DestinationOrderQosPolicyImpl( DestinationOrderQosPolicyKind.    ↙
    BY_RECEPTION_TIMESTAMP,this.GetBootstrap());
144                         break;
145                     case DestinationOrder.BY_SOURCE_TIMESTAMP:
146                         dpqMod =new DestinationOrderQosPolicyImpl( DestinationOrderQosPolicyKind.    ↙
    BY_SOURCE_TIMESTAMP,this.GetBootstrap());
147                         break;
148
149
150                 }
151
152                 dataWriterqos.DestinationOrder = dpqMod;
153             }
154             if (dataWriterProfileQos.History != null)
155             {
156                 // TODO dataWriterProfileQos.History
157                 HistoryQosPolicyImpl dpqMod = new HistoryQosPolicyImpl(this.GetBootstrap());
158                 switch (dataWriterProfileQos.History.Kind)
159                 {
160                     case History.KEEP_ALL:
161                         dpqMod.KindQos= HistoryQosPolicyKind.KEEP_ALL;
162                         break;
163                     case History.KEEP_LAST:
164                         dpqMod.KindQos = HistoryQosPolicyKind.KEEP_LAST;
165                         break;
166                 }
167                 dataWriterqos.History = dpqMod;
168             }
169             if (dataWriterProfileQos.ResourceLimits != null)
170             {
171                 // TODO dataWriterProfileQos.ResourceLimits
172                 ResourceLimitsQosPolicyImpl dpqMod = new ResourceLimitsQosPolicyImpl(this.           ↙
    GetBootstrap());
173
174             }
175             if (dataWriterProfileQos.UserData != null)
176             {
177                 // TODO dataWriterProfileQos.UserData
178                 UserDataQosPolicyImpl dpqMod = new UserDataQosPolicyImpl(this.GetBootstrap());
179
180             }
181             if (dataWriterProfileQos.Ownership != null)
182             {
183                 // TODO dataWriterProfileQos.Ownership
184                 OwnershipQosPolicyImpl dpqMod = new OwnershipQosPolicyImpl(this.GetBootstrap());
185                 switch(dataWriterProfileQos.Ownership.Kind)
186                 {
187                     case Ownership.EXCLUSIVE:
188                         dpqMod = new OwnershipQosPolicyImpl(OwnershipQosPolicyKind.EXCLUSIVE,this.   ↙
    GetBootstrap());
189                         break;
190                     case Ownership.SHARED:
191                         dpqMod =new OwnershipQosPolicyImpl( OwnershipQosPolicyKind.SHARED,this.      ↙
    GetBootstrap());
192                         break;
193
194                 }
195
196
197
198             }
```

```
199                 if (dataWriterProfileQos.Lifespan != null)
200                 {
201                     LifespanQosPolicyImpl dpqMod = new LifespanQosPolicyImpl(this.GetBootstrap());
202 
203                 }
204                 if (dataWriterProfileQos.OwnershipStrength != null)
205                 {
206                     OwnershipStrengthQosPolicyImpl dpqMod = new OwnershipStrengthQosPolicyImpl(this.
    GetBootstrap());
207 
208                 }
209 
210             }
211 
212         }
213 
214         public DataWriter<TYPE> CreateDataWriter<TYPE>(Topic<TYPE> topic)
215         {
216             DataWriter<TYPE> dw = null;
217             dw = new DataWriterImpl<TYPE>(this, topic);
218             datawriters.Add(dw);
219             return dw;
220         }
221 
222         public DataWriter<TYPE> CreateDataWriter<TYPE>(Topic<TYPE> topic, DataWriterQos qos,
    DataWriterListener<TYPE> listener, ICollection<Type> statuses)
223         {
224             DataWriter<TYPE> dw = null;
225             dw = new DataWriterImpl<TYPE>(this, topic, qos, listener, statuses);
226             datawriters.Add(dw);
227             return dw;
228         }
229 
230         public DataWriter<TYPE> CreateDataWriter<TYPE>(Topic<TYPE> topic, string qosLibraryName, string
    qosProfileName, DataWriterListener<TYPE> listener, ICollection<Type> statuses)
231         {
232             throw new NotImplementedException();
233         }
234 
235         public org.omg.dds.type.builtin.BytesDataWriter CreateBytesDataWriter(Topic<byte[]> topic)
236         {
237             throw new NotImplementedException();
238         }
239 
240         public org.omg.dds.type.builtin.BytesDataWriter CreateBytesDataWriter(Topic<byte[]> topic,
    DataWriterQos qos, DataWriterListener<byte[]> listener, ICollection<Type> statuses)
241         {
242             throw new NotImplementedException();
243         }
244 
245         public org.omg.dds.type.builtin.BytesDataWriter CreateBytesDataWriter(Topic<byte[]> topic, string
     qosLibraryName, string qosProfileName, DataWriterListener<byte[]> listener, ICollection<Type>
    statuses)
246         {
247             throw new NotImplementedException();
248         }
249 
250         public org.omg.dds.type.builtin.KeyedBytesDataWriter CreateKeyedBytesDataWriter(org.omg.dds.topic
    .Topic<org.omg.dds.type.builtin.KeyedBytes> topic)
251         {
252             throw new NotImplementedException();
253         }
254 
255         public org.omg.dds.type.builtin.KeyedBytesDataWriter CreateKeyedBytesDataWriter(org.omg.dds.topic
    .Topic<org.omg.dds.type.builtin.KeyedBytes> topic, DataWriterQos qos, DataWriterListener<org.omg.dds.
    type.builtin.KeyedBytes> listener, ICollection<Type> statuses)
256         {
257             throw new NotImplementedException();
258         }
259 
260         public org.omg.dds.type.builtin.KeyedBytesDataWriter CreateKeyedBytesDataWriter(org.omg.dds.topic
    .Topic<org.omg.dds.type.builtin.KeyedBytes> topic, string qosLibraryName, string qosProfileName,
    DataWriterListener<org.omg.dds.type.builtin.KeyedBytes> listener, ICollection<Type> statuses)
261         {
```

```
262             throw new NotImplementedException();
263         }
264 
265         public org.omg.dds.type.builtin.stringDataWriter CreatestringDataWriter(org.omg.dds.topic.Topic
    <string> topic)
266         {
267             throw new NotImplementedException();
268         }
269 
270         public org.omg.dds.type.builtin.stringDataWriter CreatestringDataWriter(org.omg.dds.topic.Topic
    <string> topic, DataWriterQos qos, DataWriterListener<string> listener, ICollection<Type> statuses)
271         {
272             throw new NotImplementedException();
273         }
274 
275         public org.omg.dds.type.builtin.stringDataWriter CreatestringDataWriter(org.omg.dds.topic.Topic
    <string> topic, string qosLibraryName, string qosProfileName, DataWriterListener<string> listener,
    ICollection<Type> statuses)
276         {
277             throw new NotImplementedException();
278         }
279 
280         public org.omg.dds.type.builtin.KeyedstringDataWriter CreateKeyedstringDataWriter(org.omg.dds.
    topic.Topic<org.omg.dds.type.builtin.Keyedstring> topic)
281         {
282             throw new NotImplementedException();
283         }
284 
285         public org.omg.dds.type.builtin.KeyedstringDataWriter CreateKeyedstringDataWriter(org.omg.dds.
    topic.Topic<org.omg.dds.type.builtin.Keyedstring> topic, DataWriterQos qos, DataWriterListener<org.
    omg.dds.type.builtin.Keyedstring> listener, ICollection<Type> statuses)
286         {
287             throw new NotImplementedException();
288         }
289 
290         public org.omg.dds.type.builtin.KeyedstringDataWriter CreateKeyedstringDataWriter(org.omg.dds.
    topic.Topic<org.omg.dds.type.builtin.Keyedstring> topic, string qosLibraryName, string qosProfileName
    , DataWriterListener<org.omg.dds.type.builtin.Keyedstring> listener, ICollection<Type> statuses)
291         {
292             throw new NotImplementedException();
293         }
294 
295         public DataWriter<TYPE> LookupDataWriter<TYPE>(string topicName)
296         {
297             throw new NotImplementedException();
298         }
299 
300         public DataWriter<TYPE> LookupDataWriter<TYPE>(org.omg.dds.topic.Topic<TYPE> topicName)
301         {
302             throw new NotImplementedException();
303         }
304 
305         public org.omg.dds.type.builtin.BytesDataWriter LookupBytesDataWriter(org.omg.dds.topic.Topic
    <byte[]> topicName)
306         {
307             throw new NotImplementedException();
308         }
309 
310         public org.omg.dds.type.builtin.KeyedBytesDataWriter LookupKeyedBytesDataWriter(org.omg.dds.topic
    .Topic<org.omg.dds.type.builtin.KeyedBytes> topicName)
311         {
312             throw new NotImplementedException();
313         }
314 
315         public org.omg.dds.type.builtin.stringDataWriter LookupstringDataWriter(org.omg.dds.topic.Topic
    <string> topicName)
316         {
317             throw new NotImplementedException();
318         }
319 
320         public org.omg.dds.type.builtin.KeyedstringDataWriter LookupKeyedstringDataWriter(org.omg.dds.
    topic.Topic<org.omg.dds.type.builtin.Keyedstring> topicName)
321         {
322             throw new NotImplementedException();
```

```
323         }
324
325         public void CloseContainedEntities()
326         {
327             throw new NotImplementedException();
328         }
329
330         public void SuspendPublications()
331         {
332             throw new NotImplementedException();
333         }
334
335         public void ResumePublications()
336         {
337             throw new NotImplementedException();
338         }
339
340         public void BeginCoherentChanges()
341         {
342             throw new NotImplementedException();
343         }
344
345         public void EndCoherentChanges()
346         {
347             throw new NotImplementedException();
348         }
349
350         public void WaitForAcknowledgments(org.omg.dds.core.Duration maxWait)
351         {
352             throw new NotImplementedException();
353         }
354
355         public void WaitForAcknowledgments(long maxWait, TimeUnit unit)
356         {
357             throw new NotImplementedException();
358         }
359
360         public DataWriterQos GetDefaultDataWriterQos()
361         {
362             return this.dataWriterqos;
363         }
364
365         public void SetDefaultDataWriterQos(DataWriterQos qos)
366         {
367             throw new NotImplementedException();
368         }
369
370         public void SetDefaultDataWriterQos(string qosLibraryName, string qosProfileName)
371         {
372             throw new NotImplementedException();
373         }
374
375         public void CopyFromTopicQos(DataWriterQos dst, org.omg.dds.topic.TopicQos src)
376         {
377             throw new NotImplementedException();
378         }
379
380         public DomainParticipant GetParent()
381         {
382             return this.parent;
383         }
384
385         public PublisherListener GetListener()
386         {
387             return this.listener;
388         }
389
390         public void SetListener(PublisherListener listener)
391         {
392             this.listener = listener;
393         }
394
395         public PublisherQos GetQos()
396         {
```

```
397             return this.qos;
398         }
399 
400         public void SetQos(PublisherQos qos)
401         {
402             this.qos = qos;
403         }
404 
405         public void SetQos(string qosLibraryName, string qosProfileName)
406         {
407             throw new NotImplementedException();
408         }
409 
410         public void Enable()
411         {
412             throw new NotImplementedException();
413         }
414 
415         public org.omg.dds.core.StatusCondition<Publisher> GetStatusCondition()
416         {
417             throw new NotImplementedException();
418         }
419 
420         public ICollection<TYPE> GetStatusChanges<TYPE>(ICollection<TYPE> statuses)
421         {
422             throw new NotImplementedException();
423         }
424 
425         public org.omg.dds.core.InstanceHandle GetInstanceHandle()
426         {
427             throw new NotImplementedException();
428         }
429 
430         public void Close()
431         {
432             throw new NotImplementedException();
433         }
434 
435         public void Retain()
436         {
437             throw new NotImplementedException();
438         }
439 
440         public Bootstrap GetBootstrap()
441         {
442             return Bootstrap;
443         }
444     }
445 }
446 
```

```
using Doopec.Dds.Core.Policy;
using Doopec.DDS.Core;
using Doopec.DDS.Core.Policy;
using org.omg.dds.core;
using org.omg.dds.core.policy;
using org.omg.dds.pub;
using org.omg.dds.pub.modifiable;
using System;

namespace Doopec.Dds.Pub
{
    public class PublisherQosImpl : EntityQosImpl<PublisherQos, ModifiablePublisherQos>, PublisherQos
    {
        private PresentationQosPolicy presentationQosPolicy;
        private PartitionQosPolicy partitionQosPolicy;
        private GroupDataQosPolicy groupDataQosPolicy;
        private EntityFactoryQosPolicy entityFactoryQosPolicy;

        public PublisherQosImpl(Bootstrap boostrap)
            : base(boostrap)
        {
            presentationQosPolicy = new PresentationQosPolicyImpl(this.GetBootstrap());
            partitionQosPolicy = new PartitionQosPolicyImpl(this.GetBootstrap());
            groupDataQosPolicy = new GroupDataQosPolicyImpl(this.GetBootstrap());
            entityFactoryQosPolicy = new EntityFactoryQosPolicyImpl(this.GetBootstrap());

        }
        public PresentationQosPolicy GetPresentation()
        {
            return presentationQosPolicy;
        }

        public PartitionQosPolicy GetPartition()
        {
            return partitionQosPolicy;
        }

        public GroupDataQosPolicy GetGroupData()
        {
            return groupDataQosPolicy;
        }

        public EntityFactoryQosPolicy GetEntityFactory()
        {
            return entityFactoryQosPolicy;
        }

        public new System.Collections.IEnumerator GetEnumerator()
        {
            throw new NotImplementedException();
        }

        public override ModifiablePublisherQos Modify()
        {
            throw new NotImplementedException();
        }
    }
}

```

```
1  using Doopec.DDS.Sub;
2  using org.omg.dds.core;
3  using org.omg.dds.core.policy;
4  using org.omg.dds.sub;
5  using org.omg.dds.sub.modifiable;
6  using System;
7  using System.Collections.Generic;
8  using System.Linq;
9  using System.Text;
10 using System.Threading.Tasks;
11
12 namespace Doopec.Dds.Sub.modifiable
13 {
14     class ModifiableDataReaderQosImpl : ModifiableDataReaderQos
15     {
16         private DataReaderQosImpl innerQos;
17         public ModifiableDataReaderQosImpl(DataReaderQosImpl qos)
18         {
19             this.innerQos = qos;
20         }
21
22         public ModifiableDataReaderQosImpl(Bootstrap boostrap)
23         {
24             this.innerQos = new DataReaderQosImpl(boostrap);
25         }
26         public ModifiableDataReaderQos SetTimeBasedFilter(TimeBasedFilterQosPolicy timeBasedFilter)
27         {
28             this.innerQos.TimeBasedFilter = timeBasedFilter;
29             return this;
30         }
31         public ModifiableDataReaderQos SetDurability(DurabilityQosPolicy durability)
32         {
33             this.innerQos.Durability = durability;
34             return this;
35         }
36
37         public ModifiableDataReaderQos SetDurabilityService(DurabilityServiceQosPolicy durabilityService)
38         {
39             this.innerQos.DurabilityService = durabilityService;
40             return this;
41         }
42
43         public ModifiableDataReaderQos SetDeadline(DeadlineQosPolicy deadline)
44         {
45             this.innerQos.Deadline = deadline;
46             return this;
47         }
48
49         public ModifiableDataReaderQos SetLatencyBudget(LatencyBudgetQosPolicy latencyBudget)
50         {
51             this.innerQos.LatencyBudget = latencyBudget;
52             return this;
53         }
54
55         public ModifiableDataReaderQos SetLiveliness(LivelinessQosPolicy liveliness)
56         {
57             this.innerQos.Liveliness = liveliness;
58             return this;
59         }
60
61         public ModifiableDataReaderQos SetReliability(ReliabilityQosPolicy reliability)
62         {
63             this.innerQos.Reliability = reliability;
64             return this;
65         }
66
67         public ModifiableDataReaderQos SetDestinationOrder(DestinationOrderQosPolicy destinationOrder)
68         {
69             this.innerQos.DestinationOrder = destinationOrder;
70             return this;
71         }
72
73         public ModifiableDataReaderQos SetHistory(HistoryQosPolicy history)
74         {
```

```
75              this.innerQos.History = history;
76              return this;
77          }
78
79          public ModifiableDataReaderQos SetResourceLimits(ResourceLimitsQosPolicy resourceLimits)
80          {
81              this.innerQos.ResourceLimits = resourceLimits;
82              return this;
83          }
84
85          public ModifiableDataReaderQos SetTransportPriority(TransportPriorityQosPolicy transportPriority)
86          {
87              this.innerQos.TransportPriority = transportPriority;
88              return this;
89          }
90
91          public ModifiableDataReaderQos SetLifespan(LifespanQosPolicy lifespan)
92          {
93              this.innerQos.Lifespan = lifespan;
94              return this;
95          }
96
97          public ModifiableDataReaderQos SetUserData(UserDataQosPolicy userData)
98          {
99              this.innerQos.UserData = userData;
100             return this;
101         }
102
103         public ModifiableDataReaderQos SetOwnership(OwnershipQosPolicy ownership)
104         {
105             this.innerQos.Ownership = ownership;
106             return this;
107         }
108
109         public ModifiableDataReaderQos SetOwnershipStrength(OwnershipStrengthQosPolicy ownershipStrength)
110         {
111             this.innerQos.OwnershipStrength = ownershipStrength;
112             return this;
113         }
114
115         public ModifiableDataReaderQos SetReaderDataLifecycle(ReaderDataLifecycleQosPolicy ↙
    readerDataLifecycle)
116         {
117             this.innerQos.ReaderDataLifecycle = readerDataLifecycle;
118             return this;
119         }
120
121         public ModifiableDataReaderQos SetRepresentation(DataRepresentationQosPolicy representation)
122         {
123             this.innerQos.DataRepresentation = representation;
124             return this;
125         }
126
127         public ModifiableDataReaderQos SetTypeConsistency(TypeConsistencyEnforcementQosPolicy ↙
    typeConsistency)
128         {
129             this.innerQos.TypeConsistencyEnforcement = typeConsistency;
130             return this;
131         }
132
133         public ModifiableDataReaderQos CopyFrom(org.omg.dds.topic.TopicQos src)
134         {
135             throw new NotImplementedException();
136         }
137
138         public new System.Collections.IEnumerator GetEnumerator()
139         {
140             throw new NotImplementedException();
141         }
142
143         public POLICY put<POLICY>(QosPolicyId key, POLICY value) where POLICY : QosPolicy
144         {
145             throw new NotImplementedException();
146         }
```

```
147
148         public ModifiableDataReaderQos CopyFrom(DataReaderQos other)
149         {
150             return new ModifiableDataReaderQosImpl(other.GetBootstrap());
151         }
152
153         public DataReaderQos FinishModification()
154         {
155             return new DataReaderQosImpl(this.GetBootstrap());
156         }
157
158
159
160
161         public DurabilityQosPolicy GetDurability()
162         {
163             return this.innerQos.GetDurability();
164         }
165
166         public DeadlineQosPolicy GetDeadline()
167         {
168             return this.innerQos.GetDeadline();
169         }
170
171         public LatencyBudgetQosPolicy GetLatencyBudget()
172         {
173             return this.innerQos.GetLatencyBudget();
174         }
175
176         public LivelinessQosPolicy GetLiveliness()
177         {
178             return this.innerQos.GetLiveliness();
179         }
180
181         public DestinationOrderQosPolicy GetDestinationOrder()
182         {
183             return this.innerQos.GetDestinationOrder();
184         }
185
186         public HistoryQosPolicy GetHistory()
187         {
188             return this.innerQos.GetHistory();
189         }
190
191         public ResourceLimitsQosPolicy GetResourceLimits()
192         {
193             return this.innerQos.GetResourceLimits();
194         }
195
196         public UserDataQosPolicy GetUserData()
197         {
198             return this.innerQos.GetUserData();
199         }
200
201         public OwnershipQosPolicy GetOwnership()
202         {
203             return this.innerQos.GetOwnership();
204         }
205
206         public TimeBasedFilterQosPolicy GetTimeBasedFilter()
207         {
208             return this.innerQos.GetTimeBasedFilter();
209         }
210
211         public ReliabilityQosPolicy GetReliability()
212         {
213             return this.innerQos.GetReliability();
214         }
215
216         public ReaderDataLifecycleQosPolicy GetReaderDataLifecycle()
217         {
218             return this.innerQos.GetReaderDataLifecycle();
219         }
220
```

```
221         public DataRepresentationQosPolicy GetRepresentation()
222         {
223             return this.innerQos.GetRepresentation();
224         }
225 
226         public TypeConsistencyEnforcementQosPolicy GetTypeConsistency()
227         {
228             return this.innerQos.GetTypeConsistency();
229         }
230 
231         public POLICY Get<POLICY>(QosPolicyId id) where POLICY : QosPolicy
232         {
233             throw new NotImplementedException();
234         }
235 
236         public QosPolicy Put(QosPolicyId key, QosPolicy value)
237         {
238             throw new NotImplementedException();
239         }
240 
241         public QosPolicy Remove(object key)
242         {
243             throw new NotImplementedException();
244         }
245 
246         public void Clear()
247         {
248             throw new NotImplementedException();
249         }
250 
251         public Bootstrap GetBootstrap()
252         {
253             throw new NotImplementedException();
254         }
255 
256         public void Add(QosPolicyId key, QosPolicy value)
257         {
258             throw new NotImplementedException();
259         }
260 
261         public bool ContainsKey(QosPolicyId key)
262         {
263             throw new NotImplementedException();
264         }
265 
266         public ICollection<QosPolicyId> Keys
267         {
268             get { throw new NotImplementedException(); }
269         }
270 
271         public bool Remove(QosPolicyId key)
272         {
273             throw new NotImplementedException();
274         }
275 
276         public bool TryGetValue(QosPolicyId key, out QosPolicy value)
277         {
278             throw new NotImplementedException();
279         }
280 
281         public ICollection<QosPolicy> Values
282         {
283             get { throw new NotImplementedException(); }
284         }
285 
286         public QosPolicy this[QosPolicyId key]
287         {
288             get
289             {
290                 throw new NotImplementedException();
291             }
292             set
293             {
294                 throw new NotImplementedException();
```

```
295             }
296         }
297 
298         public void Add(KeyValuePair<QosPolicyId, QosPolicy> item)
299         {
300             throw new NotImplementedException();
301         }
302 
303         public bool Contains(KeyValuePair<QosPolicyId, QosPolicy> item)
304         {
305             throw new NotImplementedException();
306         }
307 
308         public void CopyTo(KeyValuePair<QosPolicyId, QosPolicy>[] array, int arrayIndex)
309         {
310             throw new NotImplementedException();
311         }
312 
313         public int Count
314         {
315             get { throw new NotImplementedException(); }
316         }
317 
318         public bool IsReadOnly
319         {
320             get { throw new NotImplementedException(); }
321         }
322 
323         public bool Remove(KeyValuePair<QosPolicyId, QosPolicy> item)
324         {
325             throw new NotImplementedException();
326         }
327 
328         IEnumerator<KeyValuePair<QosPolicyId, QosPolicy>> IEnumerable<KeyValuePair<QosPolicyId,          ↙
    QosPolicy>>.GetEnumerator()
329         {
330             throw new NotImplementedException();
331         }
332 
333         public ModifiableDataReaderQos Modify()
334         {
335             return this;
336         }
337     }
338 }
339 
```

```
1  using DDS.ConversionUtils;
2  using Doopec.Dds.Domain;
3  using Doopec.DDS.Core;
4  using Doopec.DDS.Sub;
5  using Doopec.Rtps;
6  using Doopec.Rtps.Behavior;
7  using Doopec.Rtps.SharedMem;
8  using Doopec.Rtps.Structure;
9  using org.omg.dds.core.status;
10 using org.omg.dds.sub;
11 using org.omg.dds.topic;
12 using Rtps.Behavior;
13 using Rtps.Structure;
14 using System;
15 using System.Collections.Generic;
16 using System.Threading;
17
18 namespace Doopec.Dds.Sub
19 {
20     public class DataReaderImpl<TYPE> : DataReader<TYPE>
21     {
22         TopicDescription<TYPE> topic_;
23         Subscriber sub_;
24         DataReaderListener<TYPE> listener;
25         public readonly ManualResetEvent ManualResetEvent = new ManualResetEvent(false);
26
27         /// <summary>
28         ///
29         /// </summary>
30         /// <typeparam name="T"></typeparam>
31         /// <param name="?"></param>
32         /// <returns></returns>
33         protected readonly Reader<TYPE> rtpsReader;
34
35
36         public DataReaderImpl(Subscriber sub, TopicDescription<TYPE> topic, DataReaderQos qos,
       DataReaderListener<TYPE> listener, ICollection<Type> statuses)
37         {
38             this.sub_ = sub;
39             this.topic_ = topic;
40             this.listener = listener;
41
42             RtpsStatefulReader<TYPE> reader = new RtpsStatefulReader<TYPE>((sub.GetParent() as
       DomainParticipantImpl).ParticipantGuid);
43             reader.ReaderCache.Changed += NewMessage;
44             this.rtpsReader = reader;
45         }
46
47         public DataReaderImpl(Subscriber sub, TopicDescription<TYPE> topic)
48             : this(sub, topic, sub.GetDefaultDataReaderQos(), null, null)
49         {
50         }
51
52         public Type GetType()
53         {
54             throw new NotImplementedException();
55         }
56
57         public DataReader<OTHER> Cast<OTHER>()
58         {
59             throw new NotImplementedException();
60         }
61
62         public ReadCondition<TYPE> CreateReadCondition()
63         {
64             throw new NotImplementedException();
65         }
66
67         public ReadCondition<TYPE> CreateReadCondition(ICollection<SampleState> sampleStates, ICollection
       <ViewState> viewStates, ICollection<InstanceState> instanceStates)
68         {
69             throw new NotImplementedException();
70         }
71
```

```
72          public QueryCondition<TYPE> CreateQueryCondition(string queryExpression, List<string>
        queryParameters)
73          {
74              throw new NotImplementedException();
75          }
76
77          public QueryCondition<TYPE> CreateQueryCondition(ICollection<SampleState> sampleStates,
        ICollection<ViewState> viewStates, ICollection<InstanceState> instanceStates, string queryExpression,
         List<string> queryParameters)
78          {
79              throw new NotImplementedException();
80          }
81
82          public void CloseContainedEntities()
83          {
84              throw new NotImplementedException();
85          }
86
87          public org.omg.dds.topic.TopicDescription<TYPE> GetTopicDescription()
88          {
89              throw new NotImplementedException();
90          }
91
92          public org.omg.dds.core.status.SampleRejectedStatus<TYPE> GetSampleRejectedStatus(org.omg.dds.
        core.status.SampleRejectedStatus<TYPE> status)
93          {
94              throw new NotImplementedException();
95          }
96
97          public org.omg.dds.core.status.LivelinessChangedStatus<TYPE> GetLivelinessChangedStatus(org.omg.
        dds.core.status.LivelinessChangedStatus<TYPE> status)
98          {
99              throw new NotImplementedException();
100         }
101
102         public org.omg.dds.core.status.RequestedDeadlineMissedStatus<TYPE>
        GetRequestedDeadlineMissedStatus(org.omg.dds.core.status.RequestedDeadlineMissedStatus<TYPE> status)
103         {
104             throw new NotImplementedException();
105         }
106
107         public org.omg.dds.core.status.RequestedIncompatibleQosStatus<TYPE>
        GetRequestedIncompatibleQosStatus(org.omg.dds.core.status.RequestedIncompatibleQosStatus<TYPE>
        status)
108         {
109             throw new NotImplementedException();
110         }
111
112         public org.omg.dds.core.status.SubscriptionMatchedStatus<TYPE> GetSubscriptionMatchedStatus(org.
        omg.dds.core.status.SubscriptionMatchedStatus<TYPE> status)
113         {
114             throw new NotImplementedException();
115         }
116
117         public org.omg.dds.core.status.SampleLostStatus<TYPE> GetSampleLostStatus(org.omg.dds.core.status
        .SampleLostStatus<TYPE> status)
118         {
119             throw new NotImplementedException();
120         }
121
122         public void WaitForHistoricalData(org.omg.dds.core.Duration maxWait)
123         {
124             throw new NotImplementedException();
125         }
126         private void NewMessage(object sender, EventArgs e)
127         {
128             if (this.listener != null)
129             {
130                 DataAvailableStatus<TYPE> status = new DataAvailableStatusImpl<TYPE>(this);
131
132                 this.listener.OnDataAvailable(status);
133             }
134             ManualResetEvent.Set();
135         }
```

```
136
137         public void WaitForHistoricalData(long maxWait, TimeUnit unit)
138         {
139             long tiks = 0;
140             switch (unit)
141             {
142                 case TimeUnit.DAYS:
143                     tiks = maxWait * TimeSpan.TicksPerDay;
144                     break;
145                 case TimeUnit.HOURS:
146                     tiks = maxWait * TimeSpan.TicksPerHour;
147                     break;
148                 case TimeUnit.MILLISECONDS:
149                     tiks = maxWait * TimeSpan.TicksPerMillisecond;
150                     break;
151                 case TimeUnit.MINUTES:
152                     tiks = maxWait * TimeSpan.TicksPerMinute;
153                     break;
154                 case TimeUnit.SECONDS:
155                     tiks = maxWait * TimeSpan.TicksPerSecond;
156                     break;
157                 default:
158                     throw new NotSupportedException("Time Unit " + unit);
159             }
160             ManualResetEvent.Reset();
161             if (rtpsReader.ReaderCache.Changes.Count > 0  || ManualResetEvent.WaitOne(new TimeSpan ↙
    (tiks)))
162             {
163                 if (rtpsReader.ReaderCache.Changes.Count <= 0) return;
164                 if (this.listener != null)
165                 {
166                     DataAvailableStatus<TYPE> status = new DataAvailableStatusImpl<TYPE>(this);
167
168                     this.listener.OnDataAvailable(status);
169                 }
170             }
171         }
172
173         public ICollection<org.omg.dds.core.InstanceHandle> GetMatchedPublications(ICollection<org.omg. ↙
    dds.core.InstanceHandle> publicationHandles)
174         {
175             throw new NotImplementedException();
176         }
177
178         public org.omg.dds.topic.PublicationBuiltinTopicData GetMatchedPublicationData(org.omg.dds.topic.↙
    PublicationBuiltinTopicData publicationData, org.omg.dds.core.InstanceHandle publicationHandle)
179         {
180             throw new NotImplementedException();
181         }
182
183         public Sample<TYPE> CreateSample()
184         {
185             throw new NotImplementedException();
186         }
187
188         public SampleIterator<TYPE> Read()
189         {
190             throw new NotImplementedException();
191         }
192
193         public SampleIterator<TYPE> Read(ICollection<SampleState> sampleStates, ICollection<ViewState>  ↙
    viewStates, ICollection<InstanceState> instanceStates)
194         {
195             throw new NotImplementedException();
196         }
197
198         public void Read(IList<Sample<TYPE>> samples)
199         {
200             throw new NotImplementedException();
201         }
202
203         public void Read(IList<Sample<TYPE>> samples, int maxSamples, ICollection<SampleState>          ↙
    sampleStates, ICollection<ViewState> viewStates, ICollection<InstanceState> instanceStates)
204         {
```

```
205             throw new NotImplementedException();
206         }
207
208         public SampleIterator<TYPE> Take()
209         {
210             SampleIterator<TYPE> it = new SampleIteratorImpl<TYPE>();
211             while (rtpsReader.ReaderCache.Changes.Count > 0)
212             {
213                 CacheChange<TYPE> change = rtpsReader.ReaderCache.GetChange();
214                 it.Add(new SampleImpl<TYPE>(change));
215                 rtpsReader.ReaderCache.RemoveChange(change);
216             }
217             return it;
218         }
219
220         public SampleIterator<TYPE> Take(ICollection<SampleState> sampleStates, ICollection<ViewState>
    viewStates, ICollection<InstanceState> instanceStates)
221         {
222             throw new NotImplementedException();
223         }
224
225         public void Take(IList<Sample<TYPE>> samples)
226         {
227             throw new NotImplementedException();
228         }
229
230         public void Take(IList<Sample<TYPE>> samples, int maxSamples, ICollection<SampleState>
    sampleStates, ICollection<ViewState> viewStates, ICollection<InstanceState> instanceStates)
231         {
232             throw new NotImplementedException();
233         }
234
235         public SampleIterator<TYPE> Read(ReadCondition<TYPE> condition)
236         {
237             throw new NotImplementedException();
238         }
239
240         public void Read(IList<Sample<TYPE>> samples, ReadCondition<TYPE> condition)
241         {
242             throw new NotImplementedException();
243         }
244
245         public void Read(IList<Sample<TYPE>> samples, int maxSamples, ReadCondition<TYPE> condition)
246         {
247             throw new NotImplementedException();
248         }
249
250         public SampleIterator<TYPE> Take(ReadCondition<TYPE> condition)
251         {
252             throw new NotImplementedException();
253         }
254
255         public void Take(IList<Sample<TYPE>> samples, ReadCondition<TYPE> condition)
256         {
257             throw new NotImplementedException();
258         }
259
260         public void Take(IList<Sample<TYPE>> samples, int maxSamples, ReadCondition<TYPE> condition)
261         {
262             throw new NotImplementedException();
263         }
264
265         public bool ReadNext(Sample<TYPE> sample)
266         {
267             throw new NotImplementedException();
268         }
269
270         public bool TakeNext(Sample<TYPE> sample)
271         {
272             throw new NotImplementedException();
273         }
274
275         public SampleIterator<TYPE> Read(org.omg.dds.core.InstanceHandle handle)
276         {
```

```
277             throw new NotImplementedException();
278         }
279 
280         public SampleIterator<TYPE> Read(org.omg.dds.core.InstanceHandle handle, ICollection<SampleState>↙
     sampleStates, ICollection<ViewState> viewStates, ICollection<InstanceState> instanceStates)
281         {
282             throw new NotImplementedException();
283         }
284 
285         public void Read(IList<Sample<TYPE>> samples, org.omg.dds.core.InstanceHandle handle)
286         {
287             throw new NotImplementedException();
288         }
289 
290         public void Read(IList<Sample<TYPE>> samples, org.omg.dds.core.InstanceHandle handle, int        ↙
    maxSamples, ICollection<SampleState> sampleStates, ICollection<ViewState> viewStates, ICollection    ↙
    <InstanceState> instanceStates)
291         {
292             throw new NotImplementedException();
293         }
294 
295         public SampleIterator<TYPE> Take(org.omg.dds.core.InstanceHandle handle)
296         {
297             throw new NotImplementedException();
298         }
299 
300         public SampleIterator<TYPE> Take(org.omg.dds.core.InstanceHandle handle, ICollection<SampleState>↙
     sampleStates, ICollection<ViewState> viewStates, ICollection<InstanceState> instanceStates)
301         {
302             throw new NotImplementedException();
303         }
304 
305         public void Take(List<Sample<TYPE>> samples, org.omg.dds.core.InstanceHandle handle)
306         {
307             throw new NotImplementedException();
308         }
309 
310         public void Take(List<Sample<TYPE>> samples, org.omg.dds.core.InstanceHandle handle, int         ↙
    maxSamples, ICollection<SampleState> sampleStates, ICollection<ViewState> viewStates, ICollection    ↙
    <InstanceState> instanceStates)
311         {
312             throw new NotImplementedException();
313         }
314 
315         public SampleIterator<TYPE> ReadNext(org.omg.dds.core.InstanceHandle previousHandle)
316         {
317             throw new NotImplementedException();
318         }
319 
320         public SampleIterator<TYPE> ReadNext(org.omg.dds.core.InstanceHandle previousHandle, ICollection ↙
    <SampleState> sampleStates, ICollection<ViewState> viewStates, ICollection<InstanceState>            ↙
    instanceStates)
321         {
322             throw new NotImplementedException();
323         }
324 
325         public void ReadNext(List<Sample<TYPE>> samples, org.omg.dds.core.InstanceHandle previousHandle)
326         {
327             throw new NotImplementedException();
328         }
329 
330         public void ReadNext(List<Sample<TYPE>> samples, org.omg.dds.core.InstanceHandle previousHandle, ↙
    int maxSamples, ICollection<SampleState> sampleStates, ICollection<ViewState> viewStates, ICollection↙
    <InstanceState> instanceStates)
331         {
332             throw new NotImplementedException();
333         }
334 
335         public SampleIterator<TYPE> TakeNext(org.omg.dds.core.InstanceHandle previousHandle)
336         {
337             throw new NotImplementedException();
338         }
339 
340         public SampleIterator<TYPE> TakeNext(org.omg.dds.core.InstanceHandle previousHandle, ICollection ↙
```

```
<SampleState> sampleStates, ICollection<ViewState> viewStates, ICollection<InstanceState> instanceStates)
    {
        throw new NotImplementedException();
    }

    public void TakeNext(List<Sample<TYPE>> samples, org.omg.dds.core.InstanceHandle previousHandle)
    {
        throw new NotImplementedException();
    }

    public void TakeNext(List<Sample<TYPE>> samples, org.omg.dds.core.InstanceHandle previousHandle, int maxSamples, ICollection<SampleState> sampleStates, ICollection<ViewState> viewStates, ICollection<InstanceState> instanceStates)
    {
        throw new NotImplementedException();
    }

    public SampleIterator<TYPE> ReadNext(org.omg.dds.core.InstanceHandle previousHandle, ReadCondition<TYPE> condition)
    {
        throw new NotImplementedException();
    }

    public void ReadNext(List<Sample<TYPE>> samples, org.omg.dds.core.InstanceHandle previousHandle, ReadCondition<TYPE> condition)
    {
        throw new NotImplementedException();
    }

    public void ReadNext(List<Sample<TYPE>> samples, org.omg.dds.core.InstanceHandle previousHandle, int maxSamples, ReadCondition<TYPE> condition)
    {
        throw new NotImplementedException();
    }

    public SampleIterator<TYPE> TakeNext(org.omg.dds.core.InstanceHandle previousHandle, ReadCondition<TYPE> condition)
    {
        throw new NotImplementedException();
    }

    public void TakeNext(List<Sample<TYPE>> samples, org.omg.dds.core.InstanceHandle previousHandle, ReadCondition<TYPE> condition)
    {
        throw new NotImplementedException();
    }

    public void TakeNext(List<Sample<TYPE>> samples, org.omg.dds.core.InstanceHandle previousHandle, int maxSamples, ReadCondition<TYPE> condition)
    {
        throw new NotImplementedException();
    }

    public TYPE GetKeyValue(TYPE keyHolder, org.omg.dds.core.InstanceHandle handle)
    {
        throw new NotImplementedException();
    }

    public org.omg.dds.core.modifiable.ModifiableInstanceHandle LookupInstance(org.omg.dds.core.modifiable.ModifiableInstanceHandle handle, TYPE keyHolder)
    {
        throw new NotImplementedException();
    }

    public Subscriber GetParent()
    {
        throw new NotImplementedException();
    }

    public DataReaderListener<TYPE> GetListener()
    {
        throw new NotImplementedException();
    }
```

```
404
405         public void SetListener(DataReaderListener<TYPE> listener)
406         {
407             throw new NotImplementedException();
408         }
409
410         public DataReaderQos GetQos()
411         {
412             throw new NotImplementedException();
413         }
414
415         public void SetQos(DataReaderQos qos)
416         {
417             throw new NotImplementedException();
418         }
419
420         public void SetQos(string qosLibraryName, string qosProfileName)
421         {
422             throw new NotImplementedException();
423         }
424
425         public void Enable()
426         {
427             throw new NotImplementedException();
428         }
429
430         public org.omg.dds.core.StatusCondition<DataReader<TYPE>> GetStatusCondition()
431         {
432             throw new NotImplementedException();
433         }
434
435         public ICollection<TYPE> GetStatusChanges<TYPE>(ICollection<TYPE> statuses)
436         {
437             throw new NotImplementedException();
438         }
439
440         public org.omg.dds.core.InstanceHandle GetInstanceHandle()
441         {
442             throw new NotImplementedException();
443         }
444
445         public void Close()
446         {
447             throw new NotImplementedException();
448         }
449
450         public void Retain()
451         {
452             throw new NotImplementedException();
453         }
454
455         public org.omg.dds.core.Bootstrap GetBootstrap()
456         {
457             throw new NotImplementedException();
458         }
459     }
460 }
461
```

```
using Doopec.Dds.Sub.modifiable;
using Doopec.DDS.Core;
using org.omg.dds.core;
using org.omg.dds.core.policy;
using org.omg.dds.sub;
using org.omg.dds.sub.modifiable;
using System;

namespace Doopec.DDS.Sub
{
    public class DataReaderQosImpl : EntityQosImpl<DataReaderQos, ModifiableDataReaderQos>, DataReaderQos
    {
        internal DurabilityQosPolicy Durability { get; set; }
        internal DurabilityServiceQosPolicy DurabilityService { get; set; }
        internal DeadlineQosPolicy Deadline { get; set; }
        internal LatencyBudgetQosPolicy LatencyBudget { get; set; }
        internal LivelinessQosPolicy Liveliness { get; set; }
        internal DestinationOrderQosPolicy DestinationOrder { get; set; }
        internal HistoryQosPolicy History { get; set; }
        internal ResourceLimitsQosPolicy ResourceLimits { get; set; }
        internal UserDataQosPolicy UserData { get; set; }
        internal OwnershipQosPolicy Ownership { get; set; }
        internal OwnershipStrengthQosPolicy OwnershipStrength { get; set; }
        internal TimeBasedFilterQosPolicy TimeBasedFilter { get; set; }
        internal ReaderDataLifecycleQosPolicy ReaderDataLifecycle { get; set; }
        internal DataRepresentationQosPolicy DataRepresentation { get; set; }
        internal TypeConsistencyEnforcementQosPolicy TypeConsistencyEnforcement { get; set; }
        internal ReliabilityQosPolicy Reliability { get; set; }
        internal TransportPriorityQosPolicy TransportPriority { get; set; }
        internal LifespanQosPolicy Lifespan { get; set; }

        public DataReaderQosImpl(Bootstrap boostrap)
            : base(boostrap)
        {
        }

        public DurabilityQosPolicy GetDurability()
        {
            return Durability;
        }

        public DeadlineQosPolicy GetDeadline()
        {
            return Deadline;
        }

        public LatencyBudgetQosPolicy GetLatencyBudget()
        {
            return LatencyBudget;
        }

        public LivelinessQosPolicy GetLiveliness()
        {
            return Liveliness;
        }

        public DestinationOrderQosPolicy GetDestinationOrder()
        {
            return DestinationOrder;
        }

        public HistoryQosPolicy GetHistory()
        {
            return History;
        }

        public ResourceLimitsQosPolicy GetResourceLimits()
        {
            return ResourceLimits;
        }

        public UserDataQosPolicy GetUserData()
        {
            return UserData;
```

```
75          }
76
77          public OwnershipQosPolicy GetOwnership()
78          {
79              return Ownership;
80          }
81
82          public TimeBasedFilterQosPolicy GetTimeBasedFilter()
83          {
84              return TimeBasedFilter;
85          }
86
87          public ReaderDataLifecycleQosPolicy GetReaderDataLifecycle()
88          {
89              return ReaderDataLifecycle;
90          }
91
92          public DataRepresentationQosPolicy GetRepresentation()
93          {
94              return DataRepresentation;
95          }
96
97          public ReliabilityQosPolicy GetReliability()
98          {
99              return Reliability;
100         }
101
102         public TypeConsistencyEnforcementQosPolicy GetTypeConsistency()
103         {
104             return TypeConsistencyEnforcement;
105         }
106
107         public override ModifiableDataReaderQos Modify()
108         {
109             return new ModifiableDataReaderQosImpl(this);
110         }
111     }
112 }
113
```

```
1  using Doopec.DDS.Core;
2  using org.omg.dds.core.modifiable;
3  using org.omg.dds.sub;
4  using Rtps.Structure;
5  using System;
6  using System.Collections.Generic;
7
8  namespace Doopec.DDS.Sub
9  {
10     public class SampleImpl<TYPE> : ModifiableValueImpl<Sample<TYPE>, Sample<TYPE>>, Sample<TYPE>
11     {
12         private TYPE data;
13
14         public SampleImpl(CacheChange<TYPE> change)
15         {
16             this.data = (TYPE)change.DataValue.Value;
17         }
18
19         public TYPE GetData()
20         {
21             return data;
22         }
23
24         public SampleState GetSampleState()
25         {
26             throw new NotImplementedException();
27         }
28
29         public ViewState GetViewState()
30         {
31             throw new NotImplementedException();
32         }
33
34         public InstanceState GetInstanceState()
35         {
36             throw new NotImplementedException();
37         }
38
39         public ModifiableTime GetSourceTimestamp()
40         {
41             throw new NotImplementedException();
42         }
43
44         public ModifiableInstanceHandle GetInstanceHandle()
45         {
46             throw new NotImplementedException();
47         }
48
49         public ModifiableInstanceHandle GetPublicationHandle()
50         {
51             throw new NotImplementedException();
52         }
53
54         public int GetDisposedGenerationCount()
55         {
56             throw new NotImplementedException();
57         }
58
59         public int GetNoWritersGenerationCount()
60         {
61             throw new NotImplementedException();
62         }
63
64         public int GetSampleRank()
65         {
66             throw new NotImplementedException();
67         }
68
69         public int GetGenerationRank()
70         {
71             throw new NotImplementedException();
72         }
73
74         public int GetAbsoluteGenerationRank()
```

```
75          {
76              throw new NotImplementedException();
77          }
78      }
79
80      public class SampleIteratorImpl<IT_DATA> : List<Sample<IT_DATA>>, SampleIterator<IT_DATA>
81      {
82          public void ReturnLoan()
83          {
84              throw new NotImplementedException();
85          }
86
87          public void Remove()
88          {
89              throw new NotImplementedException();
90          }
91
92          public void Set(Sample<IT_DATA> o)
93          {
94              throw new NotImplementedException();
95          }
96      }
97  }
98
```

```
1  using Doopec.Configuration;
2  using Doopec.Dds.Core;
3  using Doopec.Dds.Core.Policy;
4  using Doopec.DDS.Core.Policy;
5  using Doopec.DDS.Sub;
6  using org.omg.dds.core;
7  using org.omg.dds.core.policy;
8  using org.omg.dds.domain;
9  using org.omg.dds.sub;
10 using org.omg.dds.topic;
11 using org.omg.dds.type.builtin;
12 using System;
13 using System.Collections;
14 using System.Collections.Generic;
15 using System.Configuration;
16
17 namespace Doopec.Dds.Sub
18 {
19     public class SubscriberImpl : Subscriber
20     {
21         private SubscriberQos qos;
22         private DomainParticipant parent;
23         private DataReaderQosImpl dataReaderqos;
24         private SubscriberListener listener;
25         private IList datawriters;
26         private DDSConfigurationSection ddsConfig = Doopec.Configuration.DDSConfigurationSection.Instance;
27         public Bootstrap Bootstrap { get; internal set; }
28
29         public SubscriberImpl(SubscriberQos qos, SubscriberListener listener, DomainParticipant dp, Bootstrap bootstrap)
30         {
31             this.qos = qos;
32             this.listener = listener;
33             this.parent = dp;
34             this.Bootstrap = bootstrap;
35             datawriters = new System.Collections.ArrayList();
36
37             dataReaderqos = new DataReaderQosImpl(this.GetBootstrap());
38 #if TODO
39             if (config.Settings["DefaultDataWriterQoS"] != null)
40             {
41                 string dataReaderqosName = config.Settings["DefaultDataReaderQoS"].Value;
42                 // TODO Assign values to dataReaderqos from configuration
43                 foreach (KeyValueElement dwqos in config.QoSDataWriterCollection)
44                 {
45                     if (dwqos.Key.Equals("durability", StringComparison.InvariantCultureIgnoreCase))
46                     {
47                         string durabilityVal = dwqos.Value;
48                     }
49                 }
50             }
51 #endif
52             string qosConfigProfile = ddsConfig.Domains[dp.DomainId].QoSProfile.Name;
53             Doopec.Configuration.Dds.QoSProfilePolicy qosProfile = ddsConfig.QoSProfiles[qosConfigProfile];
54
55             if (qosProfile != null)
56             {
57                 Doopec.Configuration.Dds.DataReaderQoS dataReaderProfileQos = qosProfile.DataReaderQoS;
58                 // TODO Assign values to dataReaderqos from configuration
59                 if (dataReaderProfileQos == null)
60                     return;
61                 if (dataReaderProfileQos.Reliability != null)
62                 {
63                     ReliabilityQosPolicyImpl dpqMod = new ReliabilityQosPolicyImpl(this.GetBootstrap());
64                     if (dataReaderProfileQos.Reliability.Kind == Doopec.Configuration.Reliability.RELIABLE)
65                         dpqMod = new ReliabilityQosPolicyImpl(ReliabilityQosPolicyKind.RELIABLE, this.GetBootstrap());
66                     else if (dataReaderProfileQos.Reliability.Kind == Doopec.Configuration.Reliability.BEST_EFFORT)
67                         dpqMod = new ReliabilityQosPolicyImpl(ReliabilityQosPolicyKind.BEST_EFFORT, this.GetBootstrap());
```

```
68                      dpqMod.MaxBlockingTimeQos = new DurationImpl(this.GetBootstrap(),
    dataReaderProfileQos.Reliability.MaxBlockingTime);
69                      dataReaderqos.Reliability = dpqMod;
70                  }
71
72                  if (dataReaderProfileQos.Durability != null)
73                  {
74                      DurabilityQosPolicyImpl dpqMod = new DurabilityQosPolicyImpl(this.GetBootstrap());
75                      switch (dataReaderProfileQos.Durability.Kind)
76                      {
77                          case Durability.PERSISTENT:
78                              dpqMod = new DurabilityQosPolicyImpl(DurabilityQosPolicyKind.PERSISTENT,this.
    GetBootstrap());
79                              break;
80                          case Durability.TRANSIENT:
81                              dpqMod= new DurabilityQosPolicyImpl( DurabilityQosPolicyKind.TRANSIENT,this.
    GetBootstrap());
82                              break;
83                          case Durability.TRANSIENT_LOCAL:
84                              dpqMod =new DurabilityQosPolicyImpl( DurabilityQosPolicyKind.TRANSIENT_LOCAL,
    this.GetBootstrap());
85                              break;
86                          case Durability.VOLATILE:
87                              dpqMod =new DurabilityQosPolicyImpl( DurabilityQosPolicyKind.VOLATILE,this.
    GetBootstrap());
88                              break;
89                      }
90
91                      dataReaderqos.Durability = dpqMod;
92                  }
93
94
95
96                  if (dataReaderProfileQos.Deadline != null)
97                  {
98                      // TODO dataWriterProfileQos.Deadline
99                      DeadlineQosPolicyImpl dpqMod = new DeadlineQosPolicyImpl(this.GetBootstrap());
100                 }
101                 if (dataReaderProfileQos.LatencyBudget != null)
102                 {
103                     // TODO dataReaderProfileQos.LatencyBudget
104                     LatencyBudgetQosPolicyImpl dpqMod = new LatencyBudgetQosPolicyImpl(this.GetBootstrap
    ());
105
106                 }
107                 if (dataReaderProfileQos.Liveliness != null)
108                 {
109                     // TODO dataReaderProfileQos.LatencyBudget
110                     LivelinessQosPolicyImpl dpqMod = new LivelinessQosPolicyImpl(this.GetBootstrap());
111                      switch (dataReaderProfileQos.Liveliness.Kind)
112                      {
113                          case Liveliness.AUTOMATIC:
114                              dpqMod.KindQos = LivelinessQosPolicyKind.AUTOMATIC;
115                              break;
116                          case Liveliness.MANUAL_BY_PARTICIPANT:
117                              dpqMod.KindQos = LivelinessQosPolicyKind.MANUAL_BY_PARTICIPANT;
118                              break;
119                          case Liveliness.MANUAL_BY_TOPIC:
120                              dpqMod.KindQos = LivelinessQosPolicyKind.MANUAL_BY_TOPIC;
121                              break;
122
123                      }
124                      dataReaderqos.Liveliness = dpqMod;
125
126                 }
127                 if (dataReaderProfileQos.DestinationOrder != null)
128                 {
129                     // TODO dataWriterProfileQos.DestinationOrder
130                     DestinationOrderQosPolicyImpl dpqMod = new DestinationOrderQosPolicyImpl(this.
    GetBootstrap());
131                     switch (dataReaderProfileQos.DestinationOrder.Kind)
132                     {
133                         case DestinationOrder.BY_RECEPTION_TIMESTAMP:
134                             dpqMod = new DestinationOrderQosPolicyImpl(DestinationOrderQosPolicyKind.
```

```
    BY_RECEPTION_TIMESTAMP, this.GetBootstrap());
135                         break;
136                     case DestinationOrder.BY_SOURCE_TIMESTAMP:
137                         dpqMod = new DestinationOrderQosPolicyImpl(DestinationOrderQosPolicyKind.
    BY_SOURCE_TIMESTAMP, this.GetBootstrap());
138                         break;
139
140
141                 }
142
143                 dataReaderqos.DestinationOrder = dpqMod;
144             }
145             if (dataReaderProfileQos.History != null)
146             {
147                 // TODO dataReaderProfileQos.History
148                 HistoryQosPolicyImpl dpqMod = new HistoryQosPolicyImpl(this.GetBootstrap());
149                 switch (dataReaderProfileQos.History.Kind)
150                 {
151                     case History.KEEP_ALL:
152                         dpqMod.KindQos = HistoryQosPolicyKind.KEEP_ALL;
153                         break;
154                     case History.KEEP_LAST:
155                         dpqMod.KindQos = HistoryQosPolicyKind.KEEP_LAST;
156                         break;
157                  }
158                 dataReaderqos.History = dpqMod;
159             }
160             if (dataReaderProfileQos.ResourceLimits != null)
161             {
162                 // TODO dataReaderProfileQos.ResourceLimits
163                 ResourceLimitsQosPolicyImpl dpqMod = new ResourceLimitsQosPolicyImpl(this.
    GetBootstrap());
164             }
165             if (dataReaderProfileQos.UserData != null)
166             {
167                 // TODO dataReaderProfileQos.UserData
168                 UserDataQosPolicyImpl dpqMod = new UserDataQosPolicyImpl(this.GetBootstrap());
169             }
170             if (dataReaderProfileQos.Ownership != null)
171             {
172                 // TODO dataReaderProfileQos.Ownership
173                 OwnershipQosPolicyImpl dpqMod = new OwnershipQosPolicyImpl(this.GetBootstrap());
174                 switch (dataReaderProfileQos.Ownership.Kind)
175                 {
176                     case Ownership.EXCLUSIVE:
177                         dpqMod.KindQos = OwnershipQosPolicyKind.EXCLUSIVE;
178                         break;
179                     case Ownership.SHARED:
180                         dpqMod.KindQos = OwnershipQosPolicyKind.SHARED;
181                         break;
182
183                 }
184                 dataReaderqos.Ownership = dpqMod;
185             }
186             if (dataReaderProfileQos.TimeBasedFilter != null)
187             {
188                 // TODO dataReaderProfileQos.TimeBasedFilter
189                 TimeBasedFilterQosPolicyImpl dpqMod = new TimeBasedFilterQosPolicyImpl(this.
    GetBootstrap());
190
191                 dpqMod.MinimumSeparationQos = new DurationImpl(this.GetBootstrap(),
    dataReaderProfileQos.TimeBasedFilter.MinimumSeparation);
192
193
194                 dataReaderqos.TimeBasedFilter = dpqMod;
195             }
196             if (dataReaderProfileQos.ReaderDataLifecycle != null)
197             {
198                 // TODO dataReaderProfileQos.ReaderDataLifecycle
199                 ReaderDataLifecycleQosPolicyImpl dpqMod = new ReaderDataLifecycleQosPolicyImpl(this.
    GetBootstrap());
200             }
201         }
202     }
```

```
203
204
205         public DataReader<TYPE> CreateDataReader<TYPE>(TopicDescription<TYPE> topic)
206         {
207             DataReader<TYPE> dw = null;
208             dw = new DataReaderImpl<TYPE>(this, topic);
209             datawriters.Add(dw);
210             return dw;
211         }
212
213         public DataReader<TYPE> CreateDataReader<TYPE>(TopicDescription<TYPE> topic, DataReaderQos qos,  ↙
    DataReaderListener<TYPE> listener, ICollection<Type> statuses)
214         {
215             DataReader<TYPE> dw = null;
216             dw = new DataReaderImpl<TYPE>(this, topic, qos, listener, statuses);
217             datawriters.Add(dw);
218             return dw;
219         }
220
221         public DataReader<TYPE> CreateDataReader<TYPE>(TopicDescription<TYPE> topic, string              ↙
    qosLibraryName, string qosProfileName, DataReaderListener<TYPE> listener, ICollection<Type> statuses)
222         {
223             throw new NotImplementedException();
224         }
225
226         public BytesDataReader CreateBytesDataReader(TopicDescription<byte[]> topic)
227         {
228             throw new NotImplementedException();
229         }
230
231         public BytesDataReader CreateBytesDataReader(TopicDescription<byte[]> topic, DataReaderQos qos,  ↙
    DataReaderListener<byte[]> listener, ICollection<Type> statuses)
232         {
233             throw new NotImplementedException();
234         }
235
236         public BytesDataReader CreateBytesDataReader(TopicDescription<byte[]> topic, string              ↙
    qosLibraryName, string qosProfileName, DataReaderListener<byte[]> listener, ICollection<Type>        ↙
    statuses)
237         {
238             throw new NotImplementedException();
239         }
240
241         public KeyedBytesDataReader CreateKeyedBytesDataReader(TopicDescription<KeyedBytes> topic)
242         {
243             throw new NotImplementedException();
244         }
245
246         public KeyedBytesDataReader CreateKeyedBytesDataReader(TopicDescription<KeyedBytes> topic,       ↙
    DataReaderQos qos, DataReaderListener<KeyedBytes> listener, ICollection<Type> statuses)
247         {
248             throw new NotImplementedException();
249         }
250
251         public KeyedBytesDataReader CreateKeyedBytesDataReader(TopicDescription<KeyedBytes> topic, string↙
     qosLibraryName, string qosProfileName, DataReaderListener<KeyedBytes> listener, ICollection<Type>   ↙
    statuses)
252         {
253             throw new NotImplementedException();
254         }
255
256         public stringDataReader CreatestringDataReader(TopicDescription<string> topic)
257         {
258             throw new NotImplementedException();
259         }
260
261         public stringDataReader CreatestringDataReader(TopicDescription<string> topic, DataReaderQos qos,↙
     DataReaderListener<string> listener, ICollection<Type> statuses)
262         {
263             throw new NotImplementedException();
264         }
265
266         public stringDataReader CreatestringDataReader(TopicDescription<string> topic, string            ↙
    qosLibraryName, string qosProfileName, DataReaderListener<string> listener, ICollection<Type>        ↙
```

```
statuses)
        {
            throw new NotImplementedException();
        }

        public KeyedstringDataReader CreateKeyedstringDataReader(TopicDescription<Keyedstring> topic)
        {
            throw new NotImplementedException();
        }

        public KeyedstringDataReader CreateKeyedstringDataReader(TopicDescription<Keyedstring> topic,
DataReaderQos qos, DataReaderListener<Keyedstring> listener, ICollection<Type> statuses)
        {
            throw new NotImplementedException();
        }

        public KeyedstringDataReader CreateKeyedstringDataReader(TopicDescription<Keyedstring> topic,
string qosLibraryName, string qosProfileName, DataReaderListener<Keyedstring> listener, ICollection
<Type> statuses)
        {
            throw new NotImplementedException();
        }

        public DataReader<TYPE> LookupDataReader<TYPE>(string topicName)
        {
            throw new NotImplementedException();
        }

        public DataReader<TYPE> LookupDataReader<TYPE>(TopicDescription<TYPE> topicName)
        {
            throw new NotImplementedException();
        }

        public BytesDataReader LookupBytesDataReader(TopicDescription<byte[]> topicName)
        {
            throw new NotImplementedException();
        }

        public KeyedBytesDataReader LookupKeyedBytesDataReader(TopicDescription<KeyedBytes> topicName)
        {
            throw new NotImplementedException();
        }

        public stringDataReader LookupstringDataReader(TopicDescription<string> topicName)
        {
            throw new NotImplementedException();
        }

        public KeyedstringDataReader LookupKeyedstringDataReader(TopicDescription<Keyedstring> topicName)
        {
            throw new NotImplementedException();
        }

        public void CloseContainedEntities()
        {
            throw new NotImplementedException();
        }

        public ICollection<DataReader<TYPE>> GetDataReaders<TYPE>(ICollection<DataReader<TYPE>> readers)
        {
            throw new NotImplementedException();
        }

        public ICollection<DataReader<TYPE>> GetDataReaders<TYPE>(ICollection<DataReader<TYPE>> readers,
ICollection<SampleState> sampleStates, ICollection<ViewState> viewStates, ICollection<InstanceState>
instanceStates)
        {
            throw new NotImplementedException();
        }

        public void NotifyDataReaders()
        {
            throw new NotImplementedException();
        }
```

```
335
336         public void BeginAccess()
337         {
338             throw new NotImplementedException();
339         }
340
341         public void EndAccess()
342         {
343             throw new NotImplementedException();
344         }
345
346         public DataReaderQos GetDefaultDataReaderQos()
347         {
348             return this.dataReaderqos;
349         }
350
351         public void SetDefaultDataReaderQos(DataReaderQos qos)
352         {
353             this.dataReaderqos = qos as DataReaderQosImpl;
354         }
355
356         public void SetDefaultDataReaderQos(string qosLibraryName, string qosProfileName)
357         {
358             throw new NotImplementedException();
359         }
360
361         public void CopyFromTopicQos(DataReaderQos dst, TopicQos src)
362         {
363             throw new NotImplementedException();
364         }
365
366         public DomainParticipant GetParent()
367         {
368             return this.parent;
369         }
370
371         public SubscriberListener GetListener()
372         {
373             return this.listener;
374         }
375
376         public void SetListener(SubscriberListener listener)
377         {
378             this.listener = listener;
379         }
380
381         public SubscriberQos GetQos()
382         {
383             return this.qos;
384         }
385
386         public void SetQos(SubscriberQos qos)
387         {
388             this.qos = qos;
389         }
390
391         public void SetQos(string qosLibraryName, string qosProfileName)
392         {
393             throw new NotImplementedException();
394         }
395
396         public void Enable()
397         {
398             throw new NotImplementedException();
399         }
400
401         public org.omg.dds.core.StatusCondition<Subscriber> GetStatusCondition()
402         {
403             throw new NotImplementedException();
404         }
405
406         public ICollection<TYPE> GetStatusChanges<TYPE>(ICollection<TYPE> statuses)
407         {
408             throw new NotImplementedException();
```

```
409         }
410
411         public org.omg.dds.core.InstanceHandle GetInstanceHandle()
412         {
413             throw new NotImplementedException();
414         }
415
416         public void Close()
417         {
418             throw new NotImplementedException();
419         }
420
421         public void Retain()
422         {
423             throw new NotImplementedException();
424         }
425
426         public Bootstrap GetBootstrap()
427         {
428             return Bootstrap;
429         }
430
431
432         public DataReader<TYPE> CreateDataReader<TYPE>(ITopicDescription topic)
433         {
434             throw new NotImplementedException();
435         }
436
437
438         public DataReader<TYPE> CreateDataReader<TYPE>(ITopicDescription topic, DataReaderQos qos,
    DataReaderListener<TYPE> listener, ICollection<Type> statuses)
439         {
440             throw new NotImplementedException();
441         }
442
443         public DataReader<TYPE> CreateDataReader<TYPE>(ITopicDescription topic, string qosLibraryName,
    string qosProfileName, DataReaderListener<TYPE> listener, ICollection<Type> statuses)
444         {
445             throw new NotImplementedException();
446         }
447     }
448 }
449
```

```
1  using DDS.ConversionUtils;
2  using Doopec.Dds.Core.Policy;
3  using org.omg.dds.core;
4  using org.omg.dds.core.policy;
5  using org.omg.dds.topic;
6  using org.omg.dds.type.typeobject;
7  using System;
8  using System.Collections.Generic;
9
10 namespace Doopec.Dds.Topic
11 {
12     public class PublicationBuiltinTopicDataImpl : PublicationBuiltinTopicData
13     {
14         protected readonly ReliabilityQosPolicy reliabilityQosPolicy;
15
16         protected BuiltinTopicKey key;
17         protected BuiltinTopicKey participantKey;
18         protected string topicName;
19         protected string typeName;
20
21         public PublicationBuiltinTopicDataImpl()
22         {
23             reliabilityQosPolicy = new ReliabilityQosPolicyImpl(ReliabilityQosPolicyKind.RELIABLE,
       Duration.NewDuration(200, TimeUnit.MILLISECONDS, null), this.GetBootstrap());
24         }
25
26         public override BuiltinTopicKey Key
27         {
28             get
29             {
30                 return key;
31             }
32         }
33
34         public override BuiltinTopicKey ParticipantKey
35         {
36             get
37             {
38                 return participantKey;
39             }
40         }
41
42         public override string TopicName
43         {
44             get
45             {
46                 return topicName;
47             }
48         }
49
50         public override string TypeName
51         {
52             get
53             {
54                 return typeName;
55             }
56         }
57
58         public override List<string> EquivalentTypeName
59         {
60             get
61             {
62                 throw new System.NotImplementedException();
63             }
64         }
65
66         public override List<string> BaseTypeName
67         {
68             get
69             {
70                 throw new System.NotImplementedException();
71             }
72         }
73
```

```
74          public override TypeObject Type
75          {
76              get
77              {
78                  throw new System.NotImplementedException();
79              }
80          }
81
82          public override DurabilityQosPolicy Durability
83          {
84              get
85              {
86                  throw new System.NotImplementedException();
87              }
88          }
89
90          public override DurabilityServiceQosPolicy DurabilityService
91          {
92              get
93              {
94                  throw new System.NotImplementedException();
95              }
96          }
97
98          public override DeadlineQosPolicy Deadline
99          {
100             get
101             {
102                 throw new System.NotImplementedException();
103             }
104         }
105
106         public override LatencyBudgetQosPolicy LatencyBudget
107         {
108             get
109             {
110                 throw new System.NotImplementedException();
111             }
112         }
113
114         public override LivelinessQosPolicy Liveliness
115         {
116             get
117             {
118                 throw new System.NotImplementedException();
119             }
120         }
121
122         public override ReliabilityQosPolicy Reliability
123         {
124             get
125             {
126                 return reliabilityQosPolicy;
127             }
128         }
129
130         public override LifespanQosPolicy Lifespan
131         {
132             get
133             {
134                 throw new System.NotImplementedException();
135             }
136         }
137
138         public override UserDataQosPolicy UserData
139         {
140             get
141             {
142                 throw new System.NotImplementedException();
143             }
144         }
145
146         public override OwnershipQosPolicy Ownership
147         {
```

```
148             get
149             {
150                 throw new System.NotImplementedException();
151             }
152         }
153 
154         public override OwnershipStrengthQosPolicy OwnershipStrength
155         {
156             get
157             {
158                 throw new System.NotImplementedException();
159             }
160         }
161 
162         public override DestinationOrderQosPolicy DestinationOrder
163         {
164             get
165             {
166                 throw new System.NotImplementedException();
167             }
168         }
169 
170         public override PresentationQosPolicy Presentation
171         {
172             get
173             {
174                 throw new System.NotImplementedException();
175             }
176         }
177 
178         public override PartitionQosPolicy Partition
179         {
180             get
181             {
182                 throw new System.NotImplementedException();
183             }
184         }
185 
186         public override TopicDataQosPolicy TopicData
187         {
188             get
189             {
190                 throw new System.NotImplementedException();
191             }
192         }
193 
194         public override GroupDataQosPolicy GroupData
195         {
196             get
197             {
198                 throw new System.NotImplementedException();
199             }
200         }
201 
202         public override DataRepresentationQosPolicy Representation
203         {
204             get
205             {
206                 throw new System.NotImplementedException();
207             }
208         }
209 
210         public override TypeConsistencyEnforcementQosPolicy TypeConsistency
211         {
212             get
213             {
214                 throw new System.NotImplementedException();
215             }
216         }
217 
218         public override PublicationBuiltinTopicData CopyFrom(PublicationBuiltinTopicData other)
219         {
220             throw new NotImplementedException();
221         }
```

```
222
223         public override PublicationBuiltinTopicData FinishModification()
224         {
225             throw new NotImplementedException();
226         }
227
228         public override PublicationBuiltinTopicData Clone()
229         {
230             throw new NotImplementedException();
231         }
232
233         public override PublicationBuiltinTopicData Modify()
234         {
235             throw new NotImplementedException();
236         }
237
238         public override org.omg.dds.core.Bootstrap GetBootstrap()
239         {
240             throw new NotImplementedException();
241         }
242     }
243 }
244
```

```
1  using DDS.ConversionUtils;
2  using Doopec.Dds.Core.Policy;
3  using org.omg.dds.core;
4  using org.omg.dds.core.policy;
5  using org.omg.dds.topic;
6  using System;
7  using System.Collections.Generic;
8
9  namespace Doopec.Dds.Topic
10 {
11     public class SubscriptionBuiltinTopicDataImpl : SubscriptionBuiltinTopicData
12     {
13         protected readonly ReliabilityQosPolicy reliabilityQosPolicy;
14
15
16         protected BuiltinTopicKey key;
17         protected BuiltinTopicKey participantKey;
18         protected string topicName;
19         protected string typeName;
20
21         public SubscriptionBuiltinTopicDataImpl()
22         {
23             reliabilityQosPolicy = new ReliabilityQosPolicyImpl(ReliabilityQosPolicyKind.RELIABLE,
       Duration.NewDuration(200, TimeUnit.MILLISECONDS, null), this.GetBootstrap());
24         }
25
26         public override BuiltinTopicKey Key
27         {
28             get
29             {
30                 return key;
31             }
32         }
33
34         public override BuiltinTopicKey ParticipantKey
35         {
36             get
37             {
38                 return participantKey;
39             }
40         }
41
42         public override string TopicName
43         {
44             get
45             {
46                 return topicName;
47             }
48         }
49
50         public override string TypeName
51         {
52             get
53             {
54                 return typeName;
55             }
56         }
57
58         public override List<string> EquivalentTypeName
59         {
60             get
61             {
62                 throw new NotImplementedException();
63             }
64         }
65
66         public override List<string> BaseTypeName
67         {
68             get
69             {
70                 throw new NotImplementedException();
71             }
72         }
73
```

```
74          public override org.omg.dds.type.typeobject.TypeObject Type
75          {
76              get
77              {
78                  throw new NotImplementedException();
79              }
80          }
81
82          public override org.omg.dds.core.policy.DurabilityQosPolicy Durability
83          {
84              get
85              {
86                  throw new NotImplementedException();
87              }
88          }
89
90          public override org.omg.dds.core.policy.DeadlineQosPolicy Deadline
91          {
92              get
93              {
94                  throw new NotImplementedException();
95              }
96          }
97
98          public override org.omg.dds.core.policy.LatencyBudgetQosPolicy LatencyBudget
99          {
100             get
101             {
102                 throw new NotImplementedException();
103             }
104         }
105
106         public override org.omg.dds.core.policy.LivelinessQosPolicy Liveliness
107         {
108             get
109             {
110                 throw new NotImplementedException();
111             }
112         }
113
114         public override org.omg.dds.core.policy.ReliabilityQosPolicy Reliability
115         {
116             get
117             {
118                 throw new NotImplementedException();
119             }
120         }
121
122         public override org.omg.dds.core.policy.OwnershipQosPolicy Ownership
123         {
124             get
125             {
126                 throw new NotImplementedException();
127             }
128         }
129
130         public override org.omg.dds.core.policy.DestinationOrderQosPolicy DestinationOrder
131         {
132             get
133             {
134                 throw new NotImplementedException();
135             }
136         }
137
138         public override org.omg.dds.core.policy.UserDataQosPolicy UserData
139         {
140             get
141             {
142                 throw new NotImplementedException();
143             }
144         }
145
146         public override org.omg.dds.core.policy.TimeBasedFilterQosPolicy TimeBasedFilter
147         {
```

```
148             get
149             {
150                 throw new NotImplementedException();
151             }
152         }
153 
154         public override org.omg.dds.core.policy.PresentationQosPolicy Presentation
155         {
156             get
157             {
158                 throw new NotImplementedException();
159             }
160         }
161 
162         public override org.omg.dds.core.policy.PartitionQosPolicy Partition
163         {
164             get
165             {
166                 throw new NotImplementedException();
167             }
168         }
169 
170         public override org.omg.dds.core.policy.TopicDataQosPolicy TopicData
171         {
172             get
173             {
174                 throw new NotImplementedException();
175             }
176         }
177 
178         public override org.omg.dds.core.policy.GroupDataQosPolicy GroupData
179         {
180             get
181             {
182                 throw new NotImplementedException();
183             }
184         }
185 
186         public override org.omg.dds.core.policy.DataRepresentationQosPolicy Representation
187         {
188             get
189             {
190                 throw new NotImplementedException();
191             }
192         }
193 
194         public override org.omg.dds.core.policy.TypeConsistencyEnforcementQosPolicy TypeConsistency
195         {
196             get
197             {
198                 throw new NotImplementedException();
199             }
200         }
201 
202         public override SubscriptionBuiltinTopicData CopyFrom(SubscriptionBuiltinTopicData other)
203         {
204             throw new NotImplementedException();
205         }
206 
207         public override SubscriptionBuiltinTopicData FinishModification()
208         {
209             throw new NotImplementedException();
210         }
211 
212         public override SubscriptionBuiltinTopicData Modify()
213         {
214             throw new NotImplementedException();
215         }
216 
217         public override org.omg.dds.core.Bootstrap GetBootstrap()
218         {
219             throw new NotImplementedException();
220         }
221     }
```

```
222 }
223
```

```
1  using DDS.ConversionUtils;
2  using Doopec.Dds.Core.Policy;
3  using org.omg.dds.core;
4  using org.omg.dds.core.policy;
5  using org.omg.dds.topic;
6  using System;
7  using System.Collections.Generic;
8
9  namespace Doopec.Dds.Topic
10 {
11     public class TopicBuiltinTopicDataImpl : TopicBuiltinTopicData
12     {
13         protected readonly ReliabilityQosPolicy reliabilityQosPolicy;
14
15         protected BuiltinTopicKey key;
16         protected string topicName;
17         protected string typeName;
18
19         public TopicBuiltinTopicDataImpl()
20         {
21             reliabilityQosPolicy = new ReliabilityQosPolicyImpl(ReliabilityQosPolicyKind.RELIABLE,
       Duration.NewDuration(200, TimeUnit.MILLISECONDS, null), this.GetBootstrap());
22         }
23
24         public override BuiltinTopicKey Key
25         {
26             get
27             {
28                 return key;
29             }
30         }
31
32         public override string Name
33         {
34             get
35             {
36                 return topicName;
37             }
38         }
39
40         public override string TypeName
41         {
42             get
43             {
44                 return typeName;
45             }
46         }
47
48         public override List<string> EquivalentTypeName
49         {
50             get
51             {
52                 throw new NotImplementedException();
53             }
54         }
55
56         public override List<string> BaseTypeName
57         {
58             get
59             {
60                 throw new NotImplementedException();
61             }
62         }
63
64         public override org.omg.dds.type.typeobject.TypeObject Type
65         {
66             get
67             {
68
69                 throw new NotImplementedException();
70             }
71         }
72
73         public override DurabilityQosPolicy Durability
```

```
74          {
75              get
76              {
77                  throw new NotImplementedException();
78              }
79          }
80
81          public override DurabilityServiceQosPolicy DurabilityService
82          {
83              get
84              {
85                  throw new NotImplementedException();
86              }
87          }
88
89          public override DeadlineQosPolicy Deadline
90          {
91              get
92              {
93                  throw new NotImplementedException();
94              }
95          }
96
97          public override LatencyBudgetQosPolicy LatencyBudget
98          {
99              get
100             {
101                 throw new NotImplementedException();
102             }
103         }
104
105         public override LivelinessQosPolicy Liveliness
106         {
107             get
108             {
109                 throw new NotImplementedException();
110             }
111         }
112
113         public override ReliabilityQosPolicy Reliability
114         {
115             get
116             {
117                 throw new NotImplementedException();
118             }
119         }
120
121         public override TransportPriorityQosPolicy TransportPriority
122         {
123             get
124             {
125                 throw new NotImplementedException();
126             }
127         }
128
129         public override LifespanQosPolicy Lifespan
130         {
131             get
132             {
133                 throw new NotImplementedException();
134             }
135         }
136
137         public override DestinationOrderQosPolicy DestinationOrder
138         {
139             get
140             {
141                 throw new NotImplementedException();
142             }
143         }
144
145         public override HistoryQosPolicy History
146         {
147             get
```

```
148             {
149                 throw new NotImplementedException();
150             }
151         }
152 
153         public override ResourceLimitsQosPolicy ResourceLimits
154         {
155             get
156             {
157                 throw new NotImplementedException();
158             }
159         }
160 
161         public override OwnershipQosPolicy Ownership
162         {
163             get
164             {
165                 throw new NotImplementedException();
166             }
167         }
168 
169         public override TopicDataQosPolicy TopicData
170         {
171             get
172             {
173                 throw new NotImplementedException();
174             }
175         }
176 
177         public override DataRepresentationQosPolicy Representation
178         {
179             get
180             {
181                 throw new NotImplementedException();
182             }
183         }
184 
185         public override TypeConsistencyEnforcementQosPolicy TypeConsistency
186         {
187             get
188             {
189                 throw new NotImplementedException();
190             }
191         }
192 
193         public override TopicBuiltinTopicData CopyFrom(TopicBuiltinTopicData other)
194         {
195             throw new NotImplementedException();
196         }
197 
198         public override TopicBuiltinTopicData FinishModification()
199         {
200             throw new NotImplementedException();
201         }
202 
203         public override TopicBuiltinTopicData Modify()
204         {
205             throw new NotImplementedException();
206         }
207 
208         public override Bootstrap GetBootstrap()
209         {
210             throw new NotImplementedException();
211         }
212     }
213 }
214 
```

```
1  using Doopec.Dds.Domain;
2  using org.omg.dds.core;
3  using org.omg.dds.core.status;
4  using org.omg.dds.domain;
5  using org.omg.dds.topic;
6  using System;
7  using System.Collections.Generic;
8
9  namespace Doopec.Dds.Topic
10 {
11     internal class TopicImpl<TYPE> : Topic<TYPE>
12     {
13         private string topicName;
14         private Type type;
15         private TopicQos qos;
16         private ITopicListener listener;
17         private DomainParticipant parent;
18
19         public TopicImpl( string topicName, TopicQos qos, ITopicListener listener, DomainParticipantImpl ↙
       participant)
20         {
21             this.type = typeof(TYPE);
22             this.topicName = topicName;
23             this.qos = qos;
24             this.listener = listener;
25             this.parent = participant;
26         }
27
28         public IInconsistentTopicStatus GetInconsistentTopicStatus(IInconsistentTopicStatus status)
29         {
30             throw new NotImplementedException();
31         }
32
33         public Type GetType()
34         {
35             return type;
36         }
37
38         public TopicDescription<OTHER> Cast<OTHER>()
39         {
40             throw new NotImplementedException();
41         }
42
43         public string GetTypeName()
44         {
45             return type.FullName;
46         }
47
48         public string GetName()
49         {
50             return topicName;
51         }
52
53         public DomainParticipant GetParent()
54         {
55             return parent;
56         }
57
58         public void Close()
59         {
60             throw new NotImplementedException();
61         }
62
63         public Bootstrap GetBootstrap()
64         {
65             throw new NotImplementedException();
66         }
67          public ITopicListener Listener
68          {
69              get { return listener; }
70              set { listener = value; }
71          }
72          /*
73           public ITopicListener GetListener()
```

```
74          {
75              return listener;
76          }
77
78          public void SetListener(ITopicListener listener)
79          {
80              this.listener = listener;
81          }*/
82
83      public TopicQos GetQos()
84      {
85          return qos;
86      }
87
88      public void SetQos(TopicQos qos)
89      {
90          this.qos = qos;
91      }
92
93      public void SetQos(string qosLibraryName, string qosProfileName)
94      {
95          throw new NotImplementedException();
96      }
97
98      public void Enable()
99      {
100         throw new NotImplementedException();
101     }
102
103     public StatusCondition<ITopic> GetStatusCondition()
104     {
105         throw new NotImplementedException();
106     }
107
108     public ICollection<TYPE> GetStatusChanges<TYPE>(ICollection<TYPE> statuses)
109     {
110         throw new NotImplementedException();
111     }
112
113     public InstanceHandle GetInstanceHandle()
114     {
115         throw new NotImplementedException();
116     }
117
118     public void Retain()
119     {
120         throw new NotImplementedException();
121     }
122
123     public string Name
124     {
125         get { return topicName; }
126         set { topicName = value; }
127     }
128
129     public Type Type
130     {
131         get { return type; }
132     }
133
134     public InconsistentTopicStatus<TYPE> GetInconsistentTopicStatus(InconsistentTopicStatus<TYPE>
    status)
135     {
136         throw new NotImplementedException();
137     }
138
139     TopicListener<TYPE> Entity<Topic<TYPE>, TopicListener<TYPE>, TopicQos>.GetListener()
140     {
141         throw new NotImplementedException();
142     }
143
144     public void SetListener(TopicListener<TYPE> listener)
145     {
146         throw new NotImplementedException();
```

```
147         }
148
149         StatusCondition<Topic<TYPE>> Entity<Topic<TYPE>, TopicListener<TYPE>, TopicQos>.
    GetStatusCondition()
150         {
151             throw new NotImplementedException();
152         }
153     }
154 }
155
```

```
1  using Doopec.Dds.Domain;
2  using Doopec.Rtps.Discovery;
3  using log4net;
4  using org.omg.dds.domain;
5  using Rtps.Structure.Types;
6  using System;
7  using System.Collections.Generic;
8  using System.Linq;
9  using System.Reflection;
10 using System.Text;
11 using System.Threading.Tasks;
12
13 namespace Doopec.Dds.Utils
14 {
15     public enum DiscoveryDomainParticipantEventReason
16     {
17         NEW_PARTICIPANT,
18         DELETED_PARTICIPANT,
19     }
20
21     public class DiscoveryDomainParticipantEventArgs : EventArgs
22     {
23         public DiscoveryDomainParticipantEventReason Reason { get; set; }
24         public DomainParticipant Participant { get; set; }
25     }
26
27     public delegate void DiscoveryDomainParticipantEventHandler(object sender,
       DiscoveryDomainParticipantEventArgs e);
28
29     public class DiscoveryService : IDisposable
30     {
31         private static readonly ILog log = LogManager.GetLogger(MethodBase.GetCurrentMethod().
       DeclaringType);
32
33         /// The collection of domain participants.
34         private IDictionary<GUID, DomainParticipant> participants = new Dictionary<GUID,
       DomainParticipant>();
35         private static Dictionary<int, SortedSet<DomainParticipantImpl>> participantsInDomains = new
       Dictionary<int, SortedSet<DomainParticipantImpl>>();
36
37         private static DiscoveryService theInstance;
38
39         public event DiscoveryDomainParticipantEventHandler ParticipantDiscovery;
40
41         static DiscoveryService()
42         {
43             if (theInstance == null)
44                 theInstance = new DiscoveryService();
45         }
46
47         private DiscoveryService()
48         {
49         }
50
51         public static DiscoveryService Instance
52         {
53             get
54             {
55                 return theInstance;
56             }
57         }
58
59         public void RegisterParticipant(DomainParticipant participant)
60         {
61             lock (this)
62             {
63                 DomainParticipantImpl part = participant as DomainParticipantImpl;
64                 if (part != null)
65                 {
66                     participants.Add(part.ParticipantGuid, part);
67                     if (!participantsInDomains.ContainsKey(part.DomainId))
68                         participantsInDomains.Add(part.DomainId, new SortedSet<DomainParticipantImpl>());
69                     participantsInDomains[part.DomainId].Add(part);
70                     NotifyParticipantChanges(DiscoveryDomainParticipantEventReason.NEW_PARTICIPANT,
```

```
participant);
                }
            }
        }

        public void UnregisterParticipant(DomainParticipant participant)
        {
            lock (this)
            {
                DomainParticipantImpl part = participant as DomainParticipantImpl;
                if (part != null)
                {
                    participants.Remove(part.ParticipantGuid);
                    participantsInDomains[part.DomainId].Remove(part);
                    if (participantsInDomains[part.DomainId].Count == 0)
                        participantsInDomains.Remove(part.DomainId);
                    NotifyParticipantChanges(DiscoveryDomainParticipantEventReason.DELETED_PARTICIPANT,
participant);
                }
            }
        }

        public DomainParticipant LookupParticipant(int domainId)
        {
            lock (this)
            {
                if (participantsInDomains.ContainsKey(domainId))
                {
                    return participantsInDomains[domainId].LastOrDefault();
                }
                else
                    return null;
            }
        }

        public IEnumerable<DomainParticipantImpl> ParticipantsInDomain(int domainId)
        {
            lock (this)
            {
                return participantsInDomains[domainId];
            }
        }

        public IEnumerable<DomainParticipant> Participants
        {
            get
            {
                lock (this)
                {
                    return participants.Values;
                }
            }
        }

        private void NotifyParticipantChanges(DiscoveryDomainParticipantEventReason reason,
DomainParticipant participant)
        {
            log.DebugFormat("The information about DomainParticipants has changed. Reason {0},
Participant {1}", reason, participant);
            if (ParticipantDiscovery != null)
            {
                DiscoveryDomainParticipantEventArgs dpea = new DiscoveryDomainParticipantEventArgs()
                {
                    Reason = reason,
                    Participant = participant

                };
                ParticipantDiscovery(this, dpea);
            }
        }

        public void Dispose()
        {
        }
```

```
141     }
142 }
143
```

```
using DDS.ConversionUtils;
using org.omg.dds.core;
using org.omg.dds.core.policy;
using org.omg.dds.domain;
using org.omg.dds.pub;
using org.omg.dds.sub;
using org.omg.dds.topic;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Doopec.DDS.Utils
{
    /// <summary>
    /// This class implements methods that verify whether a qos is
    ///        valid, consistent and changeable.
    ///
    /// valid - the values are in acceptable ranges without respect
    ///         to any other values.
    ///
    /// consistent - the values are consistent with each other.
    ///         The spec sometimes calls this "compatible" but I
    ///         this compatible should be reserved for matching
    ///         QoS of subscriptions and publications.
    ///         The spec is confusing in its inconsistency of the
    ///         use of "compatible" and "consistent".
    ///
    /// The qos supported in current implementation:
    ///         Liveliness  :   kind = AUTOMATIC
    ///         Reliability :   kind = RELIABLE | BEST_EFFORT
    ///                       max_blocking_time
    ///         History     :   kind = KEEP_ALL | KEEP_LAST
    ///                         depth > 1
    ///         RESOURCE_LIMITS : max_samples_per_instance
    ///
    /// Other than these supported qos, any qos that is different from the
    /// initial Value is invalid.
    /// </summary>
    public static class QosHelper
    {
        public static bool IsConsistent(DomainParticipantQos qos)
        {
            return true;
        }

        public static bool IsConsistent(TopicQos qos)
        {
            return true;
        }

        public static bool IsConsistent(DataWriterQos qos)
        {
            return true;
        }

        public static bool IsConsistent(PublisherQos qos)
        {
            return true;
        }

        public static bool IsConsistent(DataReaderQos qos)
        { throw new NotImplementedException(); }

        public static bool IsConsistent(SubscriberQos qos)
        {
            return true;
        }

        public static bool IsConsistent(DomainParticipantFactoryQos qos)
        {
            return true;
        }
```

```
75
76          // The spec does not have specification about the content of
77          // UserDataQosPolicy,TopicDataQosPolicy and GroupDataQosPolicy
78          // so they are valid with any Value.
79          public static bool IsValid(UserDataQosPolicy qos)
80          {
81              return true;
82          }
83
84          public static bool IsValid(TopicDataQosPolicy qos)
85          {
86              return true;
87          }
88
89          public static bool IsValid(GroupDataQosPolicy qos)
90          {
91              return true;
92          }
93
94          // All values of TRANSPORT_PRIORITY.Value are accepted.
95          public static bool IsValid(TransportPriorityQosPolicy qos)
96          {
97              return true;
98          }
99
100         public static bool IsValid(LifespanQosPolicy qos)
101         {
102             return Valid_duration(qos.GetDuration());
103         }
104
105         public static bool IsValid(DurabilityQosPolicy qos)
106         {
107             return
108               (qos.GetKind() == DurabilityQosPolicyKind.VOLATILE
109                || qos.GetKind() == DurabilityQosPolicyKind.TRANSIENT_LOCAL
110                || qos.GetKind() == DurabilityQosPolicyKind.TRANSIENT
111                || qos.GetKind() == DurabilityQosPolicyKind.PERSISTENT);
112         }
113
114         public static bool IsValid(DurabilityServiceQosPolicy qos)
115         { throw new NotImplementedException(); }
116
117         public static bool IsValid(PresentationQosPolicy qos)
118         {
119             return
120               (qos.GetAccessScope() == AccessScopeKind.INSTANCE
121                || qos.GetAccessScope() == AccessScopeKind.TOPIC
122                || qos.GetAccessScope() == AccessScopeKind.GROUP);
123         }
124
125         public static bool IsValid(DeadlineQosPolicy qos)
126         {
127             return Valid_duration(qos.GetPeriod());
128         }
129
130         public static bool Valid(LatencyBudgetQosPolicy qos)
131         {
132             return true;
133         }
134
135         public static bool IsValid(OwnershipQosPolicy qos)
136         { throw new NotImplementedException(); }
137
138         public static bool IsValid(OwnershipStrengthQosPolicy qos)
139         { throw new NotImplementedException(); }
140
141         public static bool IsValid(LivelinessQosPolicy qos)
142         { throw new NotImplementedException(); }
143
144         public static bool IsValid(TimeBasedFilterQosPolicy qos)
145         { throw new NotImplementedException(); }
146
147         public static bool IsValid(PartitionQosPolicy qos)
148         { throw new NotImplementedException(); }
```

```
149
150         public static bool IsValid(ReliabilityQosPolicy qos)
151         { throw new NotImplementedException(); }
152
153         public static bool IsValid(DestinationOrderQosPolicy qos)
154         { throw new NotImplementedException(); }
155
156         public static bool IsValid(HistoryQosPolicy qos)
157         { throw new NotImplementedException(); }
158
159         public static bool IsValid(ResourceLimitsQosPolicy qos)
160         { throw new NotImplementedException(); }
161
162         public static bool IsValid(EntityFactoryQosPolicy qos)
163         { throw new NotImplementedException(); }
164
165         public static bool IsValid(WriterDataLifecycleQosPolicy qos)
166         { throw new NotImplementedException(); }
167
168         public static bool IsValid(ReaderDataLifecycleQosPolicy qos)
169         { throw new NotImplementedException(); }
170
171         public static bool IsValid(DomainParticipantQos qos)
172         { throw new NotImplementedException(); }
173
174         public static bool IsValid(TopicQos qos)
175         { throw new NotImplementedException(); }
176
177         public static bool IsValid(DataWriterQos qos)
178         { throw new NotImplementedException(); }
179
180         public static bool IsValid(PublisherQos qos)
181         { throw new NotImplementedException(); }
182
183         public static bool IsValid(DataReaderQos qos)
184         { throw new NotImplementedException(); }
185
186         public static bool IsValid(SubscriberQos qos)
187         { throw new NotImplementedException(); }
188
189         public static bool IsValid(DomainParticipantFactoryQos qos)
190         {
191             return qos != null && qos.GetEntityFactory() != null;
192         }
193
194         public static bool IsChangeable(UserDataQosPolicy qos1,
195                                         UserDataQosPolicy qos2)
196         { throw new NotImplementedException(); }
197
198         public static bool IsChangeable(TopicDataQosPolicy qos1,
199                                         TopicDataQosPolicy qos2)
200         { throw new NotImplementedException(); }
201
202         public static bool IsChangeable(GroupDataQosPolicy qos1,
203                                         GroupDataQosPolicy qos2)
204         { throw new NotImplementedException(); }
205
206         public static bool IsChangeable(TransportPriorityQosPolicy qos1,
207                                         TransportPriorityQosPolicy qos2)
208         { throw new NotImplementedException(); }
209
210         public static bool IsChangeable(LifespanQosPolicy qos1,
211                                         LifespanQosPolicy qos2)
212         { throw new NotImplementedException(); }
213
214         public static bool changeable(DurabilityQosPolicy qos1,
215                                       DurabilityQosPolicy qos2)
216         { throw new NotImplementedException(); }
217
218         public static bool IsChangeable(DurabilityServiceQosPolicy qos1,
219                                         DurabilityServiceQosPolicy qos2)
220         { throw new NotImplementedException(); }
221
222         public static bool IsChangeable(PresentationQosPolicy qos1,
```

```
                                PresentationQosPolicy qos2)
        { throw new NotImplementedException(); }

        public static bool IsChangeable(DeadlineQosPolicy qos1,
                                DeadlineQosPolicy qos2)
        { throw new NotImplementedException(); }

        public static bool IsChangeable(LatencyBudgetQosPolicy qos1,
                                LatencyBudgetQosPolicy qos2)
        { throw new NotImplementedException(); }

        public static bool IsChangeable(OwnershipQosPolicy qos1,
                                OwnershipQosPolicy qos2)
        { throw new NotImplementedException(); }

        public static bool IsChangeable(OwnershipStrengthQosPolicy qos1,
                               OwnershipStrengthQosPolicy qos2)
        { throw new NotImplementedException(); }

        public static bool IsChangeable(LivelinessQosPolicy qos1,
                               LivelinessQosPolicy qos2)
        { throw new NotImplementedException(); }

        public static bool IsChangeable(TimeBasedFilterQosPolicy qos1,
                               TimeBasedFilterQosPolicy qos2)
        { throw new NotImplementedException(); }

        public static bool IsChangeable(PartitionQosPolicy qos1,
                               PartitionQosPolicy qos2)
        { throw new NotImplementedException(); }

        public static bool IsChangeable(ReliabilityQosPolicy qos1,
                               ReliabilityQosPolicy qos2)
        { throw new NotImplementedException(); }

        public static bool IsChangeable(DestinationOrderQosPolicy qos1,
                               DestinationOrderQosPolicy qos2)
        { throw new NotImplementedException(); }

        public static bool IsChangeable(HistoryQosPolicy qos1,
                               HistoryQosPolicy qos2)
        { throw new NotImplementedException(); }

        public static bool IsChangeable(ResourceLimitsQosPolicy qos1,
                               ResourceLimitsQosPolicy qos2)
        { throw new NotImplementedException(); }

        public static bool IsChangeable(EntityFactoryQosPolicy qos1,
                               EntityFactoryQosPolicy qos2)
        { throw new NotImplementedException(); }

        public static bool IsChangeable(WriterDataLifecycleQosPolicy qos1,
                               WriterDataLifecycleQosPolicy qos2)
        { throw new NotImplementedException(); }

        public static bool IsChangeable(ReaderDataLifecycleQosPolicy qos1,
                               ReaderDataLifecycleQosPolicy qos2)
        { throw new NotImplementedException(); }

        public static bool IsChangeable(DomainParticipantQos qos1,
                               DomainParticipantQos qos2)
        { throw new NotImplementedException(); }

        public static bool IsChangeable(TopicQos qos1,
                               TopicQos qos2)
        { throw new NotImplementedException(); }

        public static bool IsChangeable(DataWriterQos qos1,
                               DataWriterQos qos2)
        { throw new NotImplementedException(); }

        public static bool IsChangeable(PublisherQos qos1,
                               PublisherQos qos2)
        { throw new NotImplementedException(); }
```

```
297
298         public static bool IsChangeable(DataReaderQos qos1,
299                                 DataReaderQos qos2)
300         { throw new NotImplementedException(); }
301
302         public static bool IsChangeable(SubscriberQos qos1,
303                                 SubscriberQos qos2)
304         { throw new NotImplementedException(); }
305
306
307         /// <summary>
308         /// ENTITY_FACTORY policy is mutable. A change in the policy affects only the entities created ↙
    after the change;
309         /// not the previously created entities.
310         /// </summary>
311         /// <param name="qos1"></param>
312         /// <param name="qos2"></param>
313         /// <returns></returns>
314         public static bool IsChangeable(DomainParticipantFactoryQos qos1,
315                                         DomainParticipantFactoryQos qos2)
316         {
317             return true;
318         }
319
320
321         private static bool Valid_duration(Duration t)
322         {
323
324             // Only accept infinite or positive finite durations.  (Zero
325             // excluded).
326             //
327             // Note that it doesn't make much sense for users to Set
328             // durations less than 10 milliseconds since the underlying
329             // timer resolution is generally no better than that.
330             return t.IsInfinite() || t.GetDuration(TimeUnit.MILLISECONDS) > 0;
331         }
332     }
333 }
334
```

```
1  using org.omg.dds.type.typeobject;
2  using System;
3  using System.Collections.Generic;
4  using System.Linq;
5  using System.Text;
6  using System.Threading.Tasks;
7
8  namespace Doopec.Dds.XTypes
9  {
10     public class EnumeratedConstantImpl : EnumeratedConstant
11     {
12         public int Value { get; set; }
13         public string Name { get; set; }
14
15         public EnumeratedConstant SetValue(int value)
16         {
17             this.Value = value;
18             return this;
19         }
20
21         public int GetValue()
22         {
23             return this.Value;
24         }
25
26         public EnumeratedConstant SetName(string name)
27         {
28             this.Name = name;
29             return this;
30         }
31
32         public string GetName()
33         {
34             return this.Name;
35         }
36
37         public EnumeratedConstant CopyFrom(EnumeratedConstant other)
38         {
39             throw new NotImplementedException();
40         }
41
42         public EnumeratedConstant FinishModification()
43         {
44             throw new NotImplementedException();
45         }
46
47         public EnumeratedConstant Modify()
48         {
49             throw new NotImplementedException();
50         }
51
52         public org.omg.dds.core.Bootstrap GetBootstrap()
53         {
54             throw new NotImplementedException();
55         }
56     }
57 }
58
```

```
1  using org.omg.dds.type.typeobject;
2  using System;
3  using System.Collections.Generic;
4  using System.Linq;
5  using System.Text;
6  using System.Threading.Tasks;
7
8  namespace Doopec.Dds.XTypes
9  {
10     public class EnumerationTypeImpl : TypeImpl, EnumerationType
11     {
12         public int BitBound { get; set; }
13
14         public IList<EnumeratedConstant> Constants { get; set; }
15
16         public int GetBitBound()
17         {
18             return this.BitBound;
19         }
20
21         public EnumerationType SetBitBound(int newBitBound)
22         {
23             this.BitBound = newBitBound;
24             return this;
25         }
26
27         public IList<EnumeratedConstant> GetConstant()
28         {
29             return this.Constants;
30         }
31
32         public EnumerationType SetConstant(IList<EnumeratedConstant> newConstant)
33         {
34             this.Constants = newConstant;
35             return this;
36         }
37     }
38 }
39
```

```
using org.omg.dds.type.typeobject;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Doopec.Dds.XTypes
{
    public class MemberImpl : Member
    {
        public MemberProperty MemberProperty { get; set; }
        public IList<AnnotationUsage> AnnotationList { get; set; }

        public MemberProperty GetProperty()
        {
            return this.MemberProperty;
        }

        public Member SetProperty(MemberProperty newProperty)
        {
            this.MemberProperty = newProperty;
            return this;
        }

        public IList<AnnotationUsage> GetAnnotation()
        {
            return this.AnnotationList;
        }

        public Member SetAnnotation(IList<AnnotationUsage> newAnnotation)
        {
            this.AnnotationList = newAnnotation;
            return this;
        }

        public Member CopyFrom(Member other)
        {
            throw new NotImplementedException();
        }

        public Member FinishModification()
        {
            throw new NotImplementedException();
        }

        public Member Modify()
        {
            throw new NotImplementedException();
        }

        public org.omg.dds.core.Bootstrap GetBootstrap()
        {
            throw new NotImplementedException();
        }
    }
}

```

```
using org.omg.dds.type.typeobject;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Doopec.Dds.XTypes
{
    public class MemberPropertyImpl : MemberProperty
    {
        public MemberFlag Flag { get; set; }
        public uint MemberId { get; set; }
        public int Type { get; set; }
        public string Name { get; set; }
        public bool IsProperty { get; set; }

        public MemberProperty SetFlag(MemberFlag flag)
        {
            this.Flag = flag;
            return this;
        }

        public MemberProperty SetMemberId(uint memberId)
        {
            this.MemberId = memberId;
            return this;
        }

        public MemberProperty SetType(int type)
        {
            this.Type = type;
            return this;
        }

        public MemberProperty SetName(string name)
        {
            this.Name = name;
            return this;
        }

        public MemberProperty CopyFrom(MemberProperty other)
        {
            throw new NotImplementedException();
        }

        public MemberProperty FinishModification()
        {
            throw new NotImplementedException();
        }

        public MemberProperty Modify()
        {
            throw new NotImplementedException();
        }

        public org.omg.dds.core.Bootstrap GetBootstrap()
        {
            throw new NotImplementedException();
        }
    }
}

```

```
using org.omg.dds.type.typeobject;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Doopec.Dds.XTypes
{
    public class StructureTypeImpl : TypeImpl, StructureType
    {
        public int BaseType { get; set; }
        public IList<Member> MemberList { get; set; }

        public int GetBaseType()
        {
            return this.BaseType;
        }

        public StructureType SetBaseType(int newBaseTypeId)
        {
            this.BaseType= newBaseTypeId;
            return this;
        }

        public IList<Member> GetMember()
        {
            return this.MemberList;
        }

        public StructureType SetMember(IList<Member> newMember)
        {
            this.MemberList = newMember;
            return this;
        }
    }
}

```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using org.omg.dds.type.typeobject;

namespace Doopec.Dds.XTypes
{
    public class TypeImpl : org.omg.dds.type.typeobject.Type
    {
        public TypeProperty Property { get; set; }

        public TypeProperty GetProperty()
        {
            return this.Property;
        }

        public org.omg.dds.type.typeobject.Type SetProperty(TypeProperty newProperty)
        {
            this.Property = newProperty;
            return this;
        }

        public IList<AnnotationUsage> GetAnnotation()
        {
            throw new NotImplementedException();
        }

        public org.omg.dds.type.typeobject.Type SetAnnotation(IList<AnnotationUsage> newAnnotation)
        {
            throw new NotImplementedException();
        }

        public org.omg.dds.type.typeobject.Type CopyFrom(org.omg.dds.type.typeobject.Type other)
        {
            throw new NotImplementedException();
        }

        public org.omg.dds.type.typeobject.Type FinishModification()
        {
            throw new NotImplementedException();
        }

        public org.omg.dds.type.typeobject.Type Modify()
        {
            throw new NotImplementedException();
        }

        public org.omg.dds.core.Bootstrap GetBootstrap()
        {
            throw new NotImplementedException();
        }
    }
}

```

```
using org.omg.dds.type.typeobject;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Doopec.Dds.XTypes
{
    public class TypeObjectImpl : TypeObject
    {
        private TypeLibrary Library { get; set; }
        private int TypeId { get; set; }

        public TypeObject SetLibrary(TypeLibrary library)
        {
            Library = library;
            throw new NotImplementedException();
        }

        public TypeLibrary GetLibrary()
        {
            return Library;
        }

        public TypeObject SetTheType(int the_type)
        {
            TypeId = the_type;
            throw new NotImplementedException();
        }

        public int GetTheType()
        {
            return TypeId;
        }

        public TypeObject CopyFrom(TypeObject other)
        {
            throw new NotImplementedException();
        }

        public TypeObject FinishModification()
        {
            throw new NotImplementedException();
        }

        public TypeObject Modify()
        {
            throw new NotImplementedException();
        }

        public org.omg.dds.core.Bootstrap GetBootstrap()
        {
            throw new NotImplementedException();
        }
    }
}

```

```
using org.omg.dds.type.typeobject;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Doopec.Dds.XTypes
{
    public class TypePropertyImpl : TypeProperty
    {
        public TypeFlag Flag { get; set; }
        public int TypeId { get; set; }
        public string Name { get; set; }

        public TypeProperty SetFlag(TypeFlag flag)
        {
            this.Flag = flag;
            return this;
        }

        public TypeFlag GetFlag()
        {
            return this.Flag;
        }

        public TypeProperty SetTypeId(int typeId)
        {
            this.TypeId = typeId;
            return this;
        }

        public int GetTypeId()
        {
            return this.TypeId;
        }

        public TypeProperty SetName(string name)
        {
            this.Name = name;
            return this;
        }

        public string GetName()
        {
            return this.Name;
        }

        public TypeProperty CopyFrom(TypeProperty other)
        {
            throw new NotImplementedException();
        }

        public TypeProperty FinishModification()
        {
            throw new NotImplementedException();
        }

        public TypeProperty Modify()
        {
            throw new NotImplementedException();
        }

        public org.omg.dds.core.Bootstrap GetBootstrap()
        {
            throw new NotImplementedException();
        }
    }
}

```

```
1  using Mina.Core.Buffer;
2
3  namespace Doopec.Utils.Network.Encoders
4  {
5      public static class BufferUtils
6      {
7          /// <summary>
8          /// Aligns this buffer to given byteBoundary.
9          /// </summary>
10         /// <param name="buffer"></param>
11         /// <param name="byteBoundary"></param>
12         public static void Align(this IoBuffer buffer, int byteBoundary)
13         {
14             int position = buffer.Position;
15             int adv = (position % byteBoundary);
16
17             if (adv != 0)
18             {
19                 buffer.Position = position + (byteBoundary - adv);
20             }
21         }
22      }
23 }
24
```

```
1  using Doopec.Rtps.Messages;
2  using Mina.Core.Buffer;
3  using Rtps.Messages.Submessages.Elements;
4  using System;
5  using System.Collections.Generic;
6  using System.Reflection;
7  using System.Linq;
8
9  namespace Doopec.Encoders
10 {
11     public static class DataEncoder
12     {
13         /// <summary>
14         /// Creates an instance of DataEncapsulation.
15         /// </summary>
16         /// <param name="serializedPayload"></param>
17         /// <returns></returns>
18         public static DataEncapsulation EncapsuleCDRData(this byte[] serializedPayload, ByteOrder order)
19         {
20             return new CDREncapsulation(serializedPayload, order);
21         }
22
23         public static DataEncapsulation EncapsuleParameterListData(this ParameterList parameters,
       ByteOrder order)
24         {
25             return new ParameterListEncapsulation(parameters, order);
26         }
27
28         public static DataEncapsulation EncapsuleCDRData(this IoBuffer buffer, object dataObj, ByteOrder
       order)
29         {
30             return new CDREncapsulation(buffer, dataObj, order);
31         }
32
33
34         public static DataEncapsulation EncapsuleParameterListData(this IoBuffer buffer, object dataObj,
       ByteOrder order)
35         {
36             return new ParameterListEncapsulation(buffer, dataObj, order);
37         }
38
39         public static IEnumerable<FieldInfo> GetFieldInfos(Type type)
40         {
41             var fields = type.GetFields(BindingFlags.Public | BindingFlags.NonPublic | BindingFlags.
       Instance | BindingFlags.DeclaredOnly)
42                 .Where(fi => (fi.Attributes & FieldAttributes.NotSerialized) == 0)
43                 .OrderBy(f => f.Name, StringComparer.Ordinal);
44
45             if (type.BaseType == null)
46             {
47                 return fields;
48             }
49             else
50             {
51                 var baseFields = GetFieldInfos(type.BaseType);
52                 return baseFields.Concat(fields);
53             }
54         }
55     }
56 }
57
```

```
1
2  using Rtps.Messages.Submessages.Elements;
3
4  namespace Doopec.Utils.Network.Encoders
5  {
6      /// <summary>
7      /// IEncoder is used to transform Object to/from different data encodings.
8      /// </summary>
9      /// <typeparam name="T">
10     /// Type of this Marshaller. Type is used to enforce symmetry between
11     ///            unmarshall and marshall methods.
12     ///</typeparam>
13     public interface IEncoder<T>
14     {
15         /// <summary>
16         /// Determines whether or not a key is associated with type T.
17         /// </summary>
18         bool HasKey { get; }
19
20         /// <summary>
21         /// Extracts a key from given object. If null is returned, it is assumed to
22         /// be the same as a byte array of length 0. Returned byte array can be of any length.
23         /// However, if the byte arrays length is greater than 15, it is internally converted to
24         /// a MD5 hash.
25         /// </summary>
26         /// <param name="data"></param>
27         /// <returns></returns>
28         byte[] ExtractKey(T data);
29
30         /// <summary>
31         /// Unmarshalls given DataEncapsulation to Object.
32         /// </summary>
33         /// <param name="dEnc"></param>
34         /// <returns></returns>
35         T Decode(DataEncapsulation dEnc);
36
37         /// <summary>
38         /// Marshalls given Object to DataEncapsulation
39         /// </summary>
40         /// <param name="data"></param>
41         /// <returns></returns>
42         DataEncapsulation Encode(T data);
43     }
44 }
45
```

```
1  using log4net;
2  using Doopec.Rtps.RtpsTransport;
3  using Doopec.Rtps.SharedMem;
4  using Rtps.Behavior;
5  using Rtps.Structure;
6  using System;
7  using System.Collections.Generic;
8  using System.Linq;
9  using System.Reflection;
10 using System.Text;
11 using System.Threading.Tasks;
12 using Rtps.Structure.Types;
13
14 namespace Doopec.Rtps.Behavior
15 {
16     public class FakeRtpsReader<T> : StatefulReader<T>, IDisposable
17     {
18         private static readonly ILog log = LogManager.GetLogger(MethodBase.GetCurrentMethod().
       DeclaringType);
19
20         private IList<Writer<T>> writers = new List<Writer<T>>();
21
22         public FakeRtpsReader(GUID guid)
23             : base(guid)
24         {
25             IRtpsDiscovery discoveryModule =  RtpsEngineFactory.Instance.DiscoveryModule;
26             discoveryModule.RegisterEndpoint(this);
27             discoveryModule.EndpointDiscovery += OnDiscoveryEndpoints;
28             AddWriters(discoveryModule);
29         }
30
31         public void Dispose()
32         {
33             RemoveAllWriters();
34             IRtpsDiscovery discoveryModule = RtpsEngineFactory.Instance.DiscoveryModule;
35             discoveryModule.UnregisterEndpoint(this);
36             discoveryModule.EndpointDiscovery -= OnDiscoveryEndpoints;
37         }
38
39         private void OnDiscoveryEndpoints(object sender, DiscoveryEventArgs e)
40         {
41             Writer<T> writer = e.EventData as Writer<T>;
42             if (writer == null)
43                 return;
44             if (e.Reason == EventReason.NEW_ENDPOINT)
45                 writers.Add(writer);
46             else if (e.Reason == EventReason.NEW_ENDPOINT)
47                 writers.Remove(writer);
48         }
49
50         private void AddWriter(Writer<T> writer)
51         {
52             //TODO
53             //WriterProxy<T> writerProxy = new WriterProxy<T>();
54             //this.MatchedWriterAdd(writerProxy);
55
56             writers.Add(writer);
57             writer.HistoryCache.Changed += OnChangedHistoryCache;
58         }
59
60         private void AddWriters(IRtpsDiscovery discoveryModule)
61         {
62             foreach (var endpoint in discoveryModule.Endpoints)
63             {
64                 if (endpoint is Writer<T>)
65                     AddWriter(endpoint as Writer<T>);
66             }
67         }
68
69         private void RemoveAllWriters()
70         {
71             foreach(var writer in writers)
72                 writer.HistoryCache.Changed -= OnChangedHistoryCache;
73
```

```
74              writers.Clear();
75          }
76
77          private void OnChangedHistoryCache(object sender, EventArgs e)
78          {
79              log.Debug("A new change has been detected");
80              HistoryCache<T> whc = sender as HistoryCache<T>;
81              if (whc != null)
82              {
83                  CacheChange<T> change = whc.GetChange();
84                  ReaderCache.AddChange(change);
85                  whc.RemoveChange(change);
86              }
87          }
88      }
89  }
90
```

```
1  using Doopec.Rtps.RtpsTransport;
2  using Doopec.Rtps.SharedMem;
3  using Rtps.Behavior;
4  using Rtps.Structure;
5  using Rtps.Structure.Types;
6  using System;
7  using System.Collections.Generic;
8  using System.Linq;
9  using System.Text;
10 using System.Threading.Tasks;
11
12 namespace Doopec.Rtps.Behavior
13 {
14     public class FakeRtpsWriter<T> : StatefulWriter<T>, IDisposable
15     {
16         private IList<Reader<T>> readers = new List<Reader<T>>();
17         private WriterWorker worker;
18
19         public FakeRtpsWriter(GUID guid)
20             : base(guid)
21         {
22             IRtpsDiscovery discoveryModule =  RtpsEngineFactory.Instance.DiscoveryModule;
23             discoveryModule.RegisterEndpoint(this);
24             discoveryModule.EndpointDiscovery += OnDiscoveryEndpoints;
25             AddReaders(discoveryModule);
26
27             worker = new WriterWorker();
28             worker.Start((int)this.heartbeatPeriod.AsMillis());
29         }
30
31         public void Dispose()
32         {
33             readers.Clear();
34             IRtpsDiscovery discoveryModule = RtpsEngineFactory.Instance.DiscoveryModule;
35             discoveryModule.UnregisterEndpoint(this);
36             discoveryModule.EndpointDiscovery -= OnDiscoveryEndpoints;
37             worker.End();
38         }
39
40         private void OnDiscoveryEndpoints(object sender, DiscoveryEventArgs e)
41         {
42             Reader<T> reader = e.EventData as Reader<T>;
43             if (reader == null)
44                 return;
45              if (e.Reason == EventReason.NEW_ENDPOINT)
46                  readers.Add(reader);
47              else if (e.Reason == EventReason.NEW_ENDPOINT)
48                  readers.Remove(reader);
49         }
50
51         private void AddReader(Reader<T> writer)
52         {
53             readers.Add(writer);
54         }
55
56         private void AddReaders(IRtpsDiscovery discoveryModule)
57         {
58             foreach (var endpoint in discoveryModule.Endpoints)
59             {
60                 if (endpoint is Reader<T>)
61                     AddReader(endpoint as Reader<T>);
62             }
63         }
64     }
65 }
66
```

```
1  using log4net;
2  using Doopec.Rtps.RtpsTransport;
3  using Doopec.Rtps.SharedMem;
4  using Doopec.Utils.Transport;
5  using Mina.Core.Buffer;
6  using Rtps.Behavior;
7  using Rtps.Messages;
8  using Rtps.Messages.Types;
9  using Rtps.Structure;
10 using Rtps.Structure.Types;
11 using System;
12 using System.Collections.Generic;
13 using System.Linq;
14 using System.Reflection;
15 using System.Text;
16 using System.Threading.Tasks;
17 using Data = Rtps.Messages.Submessages.Data;
18 using DataObj = Rtps.Structure.Types.Data;
19 namespace Doopec.Rtps.Behavior
20 {
21     public class RtpsStatefulReader<T> : StatefulReader<T>, IDisposable
22     {
23         private static readonly ILog log = LogManager.GetLogger(MethodBase.GetCurrentMethod().
       DeclaringType);
24         private IList<Writer<T>> writers = new List<Writer<T>>();
25         private UDPReceiver rec;
26         public RtpsStatefulReader(GUID guid)
27             : base(guid)
28         {
29             //TODO use configuration for host and port
30             rec = new UDPReceiver(new Uri("udp://224.0.1.111:9999"), 1024);
31             rec.Start();
32             rec.MessageReceived += NewMessage;
33             IRtpsDiscovery discoveryModule = RtpsEngineFactory.Instance.DiscoveryModule;
34             discoveryModule.RegisterEndpoint(this);
35             discoveryModule.EndpointDiscovery += OnDiscoveryEndpoints;
36             AddWriters(discoveryModule);
37         }
38
39         private void NewMessage(object sender, RTPSMessageEventArgs e)
40         {
41             Message msg = e.Message;
42             log.DebugFormat("New Message has arrived from {0}", e.Session.RemoteEndPoint);
43             log.DebugFormat("Message Header: {0}", msg.Header);
44             foreach (var submsg in msg.SubMessages)
45             {
46                 switch (submsg.Kind)
47                 {
48                     case SubMessageKind.DATA:
49                         Data d = submsg as Data;
50                         log.DebugFormat("SubMessage Data: {0}", submsg.Kind);
51                         log.DebugFormat("The KeyFlag value state is: {0}", d.HasKeyFlag);
52                         log.DebugFormat("The DataFlag value state is: {0}", d.HasDataFlag);
53                         log.DebugFormat("The InlineQoSFlag value state is: {0}", d.HasInlineQosFlag);
54                         log.DebugFormat("The EndiannessFlag value state is: {0}", d.Header.Flags.
       IsLittleEndian);
55                         log.DebugFormat("The octetsToNextHeader value is: {0}", d.Header.
       SubMessageLength);
56                         log.DebugFormat("The extraFlags value is: {0}", d.ExtraFlags.Value);
57                         log.DebugFormat("The octetsToInlineQos value is: Aun no logro");
58                         log.DebugFormat("The readerID is: {0}", d.ReaderId);
59                         log.DebugFormat("The writerID is: {0}", d.WriterId);
60                         log.DebugFormat("The writerSN is: {0}", d.WriterSN);
61                         IoBuffer buf = IoBuffer.Wrap(d.SerializedPayload.DataEncapsulation.
       SerializedPayload);
62                         buf.Order = ByteOrder.LittleEndian; //(d.Header.IsLittleEndian ? ByteOrder.
       LittleEndian : ByteOrder.BigEndian);
63                         object obj = Doopec.Serializer.Serializer.Deserialize<T>(buf);
64                         CacheChange<T> change = new CacheChange<T>(ChangeKind.ALIVE, new GUID(msg.Header.
       GuidPrefix, d.WriterId), d.WriterSN, new DataObj(obj), new InstanceHandle());
65                         ReaderCache.AddChange(change);
66                         break;
67                     default:
68                         log.DebugFormat("SubMessage: {0}", submsg.Kind);
```

```
69                          break;
70                  }
71              }
72          }
73
74          public void Dispose()
75          {
76              rec.MessageReceived -= NewMessage;
77              rec.Dispose();
78
79              RemoveAllWriters();
80              IRtpsDiscovery discoveryModule = RtpsEngineFactory.Instance.DiscoveryModule;
81              discoveryModule.UnregisterEndpoint(this);
82              discoveryModule.EndpointDiscovery -= OnDiscoveryEndpoints;
83          }
84
85          private void OnDiscoveryEndpoints(object sender, DiscoveryEventArgs e)
86          {
87              Writer<T> writer = e.EventData as Writer<T>;
88              if (writer == null)
89                  return;
90              if (e.Reason == EventReason.NEW_ENDPOINT)
91                  writers.Add(writer);
92              else if (e.Reason == EventReason.NEW_ENDPOINT)
93                  writers.Remove(writer);
94          }
95
96          private void AddWriter(Writer<T> writer)
97          {
98              //TODO
99              //WriterProxy<T> writerProxy = new WriterProxy<T>();
100             //this.MatchedWriterAdd(writerProxy);
101
102             //writers.Add(writer);
103             //writer.HistoryCache.Changed += OnChangedHistoryCache;
104         }
105
106         private void AddWriters(IRtpsDiscovery discoveryModule)
107         {
108             foreach (var endpoint in discoveryModule.Endpoints)
109             {
110                 if (endpoint is Writer<T>)
111                     AddWriter(endpoint as Writer<T>);
112             }
113         }
114
115         private void RemoveAllWriters()
116         {
117             foreach (var writer in writers)
118                 writer.HistoryCache.Changed -= OnChangedHistoryCache;
119
120             writers.Clear();
121         }
122
123         private void OnChangedHistoryCache(object sender, EventArgs e)
124         {
125             log.Debug("A new change has been detected");
126             HistoryCache<T> whc = sender as HistoryCache<T>;
127             if (whc != null)
128             {
129                 CacheChange<T> change = whc.GetChange();
130                 ReaderCache.AddChange(change);
131                 whc.RemoveChange(change);
132             }
133         }
134     }
135 }
136
```

```
1  using log4net;
2  using Doopec.Encoders;
3  using Doopec.Rtps.RtpsTransport;
4  using Doopec.Rtps.SharedMem;
5  using Doopec.Serializer;
6  using Doopec.Utils.Transport;
7  using Mina.Core.Buffer;
8  using Rtps.Behavior;
9  using Rtps.Messages;
10 using Rtps.Messages.Submessages;
11 using Rtps.Messages.Submessages.Elements;
12 using Rtps.Structure;
13 using Rtps.Structure.Types;
14 using System;
15 using System.Collections.Generic;
16 using System.Reflection;
17 using Data = Rtps.Messages.Submessages.Data;
18
19 namespace Doopec.Rtps.Behavior
20 {
21     public class RtpsStatefulWriter<T> : StatefulWriter<T>, IDisposable
22     {
23         protected static readonly ILog log = LogManager.GetLogger(MethodBase.GetCurrentMethod().
       DeclaringType);
24
25         private IList<Reader<T>> readers = new List<Reader<T>>();
26         private WriterWorker worker;
27         private UDPTransmitter trans;
28         private int countaux=0;
29
30         public RtpsStatefulWriter(GUID guid)
31             : base(guid)
32         {
33             Doopec.Serializer.Serializer.Initialize(typeof(T));
34
35             IRtpsDiscovery discoveryModule = RtpsEngineFactory.Instance.DiscoveryModule;
36             discoveryModule.RegisterEndpoint(this);
37             discoveryModule.EndpointDiscovery += OnDiscoveryEndpoints;
38             AddReaders(discoveryModule);
39
40             //TODO Andres. Revisar esta direccion. Deberia venir de alguna configuracion
41             //TODO use configuration for host and port
42             trans = new UDPTransmitter(new Uri("udp://224.0.1.111:9999"), 256);
43             trans.Start();
44             worker = new WriterWorker(this.PeriodicWork);
45             worker.Start((int)this.heartbeatPeriod.AsMillis());
46         }
47
48         public void Dispose()
49         {
50             readers.Clear();
51             IRtpsDiscovery discoveryModule = RtpsEngineFactory.Instance.DiscoveryModule;
52             discoveryModule.UnregisterEndpoint(this);
53             discoveryModule.EndpointDiscovery -= OnDiscoveryEndpoints;
54             worker.End();
55             trans.Close();
56         }
57
58         private void OnDiscoveryEndpoints(object sender, DiscoveryEventArgs e)
59         {
60             Reader<T> reader = e.EventData as Reader<T>;
61             if (reader == null)
62                 return;
63             if (e.Reason == EventReason.NEW_ENDPOINT)
64                 readers.Add(reader);
65             else if (e.Reason == EventReason.NEW_ENDPOINT)
66                 readers.Remove(reader);
67         }
68
69         private void AddReader(Reader<T> writer)
70         {
71             readers.Add(writer);
72         }
73
```

```
74          private void AddReaders(IRtpsDiscovery discoveryModule)
75          {
76              foreach (var endpoint in discoveryModule.Endpoints)
77              {
78                  if (endpoint is Reader<T>)
79                      AddReader(endpoint as Reader<T>);
80              }
81          }
82          private void PeriodicWork()
83          {
84              // the RTPS Writer to repeatedly announce the availability of data by sending a Heartbeat
    Message.
85              log.DebugFormat("I have to send a Heartbeat Message,  at {0}", DateTime.Now);
86              SendHeartbeat();
87              if (HistoryCache.Changes.Count > 0)
88              {
89                  foreach (var change in HistoryCache.Changes)
90                  {
91                      //SendHeartbeat();
92                      //SendData(change);
93                      SendDataHeartbeat(change);
94                  }
95                  HistoryCache.Changes.Clear(); //TODO
96              }
97          }
98          private void SendHeartbeat()
99          {
100             // Create a Message with Heartbeat
101             Message m1 = new Message();
102
103             Heartbeat heartbeat = new Heartbeat();
104             EntityId id1 = EntityId.ENTITYID_UNKNOWN;
105             EntityId id2 = EntityId.ENTITYID_UNKNOWN;
106
107             heartbeat.readerId = id1;
108             heartbeat.writerId = id2;
109             heartbeat.firstSN = new SequenceNumber(10);
110             heartbeat.lastSN = new SequenceNumber(20);
111             heartbeat.count = 5;
112             m1.SubMessages.Add(heartbeat);
113
114             SendData(m1);
115         }
116
117         public void SendData(CacheChange<T> change)
118         {
119             // Create a Message with InfoSource
120             Message msg = new Message();
121
122             EntityId readerId = EntityId.ENTITYID_UNKNOWN;
123             EntityId writerId = change.WriterGuid.EntityId;
124             SerializedPayload payload = new SerializedPayload();
125             IoBuffer buff = IoBuffer.Allocate(1024);
126             payload.DataEncapsulation = buff.EncapsuleCDRData(change.DataValue.Value, BitConverter.
    IsLittleEndian ? ByteOrder.LittleEndian : ByteOrder.BigEndian);
127             Data data = new Data(readerId, writerId, change.SequenceNumber.LongValue, null, payload);
128             msg.SubMessages.Add(data);
129
130             // Write Message to bytes1 array
131             SendData(msg);
132         }
133
134         public void SendDataHeartbeat(CacheChange<T> change)
135         {
136             // Create a Message with InfoSource
137             Message msg = new Message();
138             EntityId readerId = EntityId.ENTITYID_UNKNOWN;
139             EntityId writerId = change.WriterGuid.EntityId;
140             SerializedPayload payload = new SerializedPayload();
141             IoBuffer buff = IoBuffer.Allocate(1024);
142             payload.DataEncapsulation = buff.EncapsuleCDRData(change.DataValue.Value, BitConverter.
    IsLittleEndian ? ByteOrder.LittleEndian : ByteOrder.BigEndian);
143             Data data = new Data(readerId, writerId, change.SequenceNumber.LongValue, null, payload);
144             msg.SubMessages.Add(data);
```

```
145
146             Heartbeat heartbeat = new Heartbeat();
147             heartbeat.readerId = readerId;
148             heartbeat.writerId = writerId;
149             heartbeat.firstSN = change.SequenceNumber;
150             heartbeat.lastSN = change.SequenceNumber;
151             heartbeat.count = 1;
152             msg.SubMessages.Add(heartbeat);
153
154             SendData(msg);
155         }
156         /*
157         public void SendInfoDestinationAckNack(CacheChange<T> change,SequenceNumberSet sns)
158         {
159
160             countaux = countaux + 1;
161             Message msg = new Message();
162             EntityId readerId = EntityId.ENTITYID_UNKNOWN;
163             EntityId writerId = change.WriterGuid.EntityId;
164             //SerializedPayload payload = new SerializedPayload();
165             IoBuffer buff = IoBuffer.Allocate(1024);
166             //payload.DataEncapsulation = buff.EncapsuleCDRData(change.DataValue.Value, BitConverter.
    IsLittleEndian ? ByteOrder.LittleEndian : ByteOrder.BigEndian);
167             InfoDestination infoDest = new InfoDestination();
168
169             msg.SubMessages.Add(infoDest);
170
171             AckNack ackNack = new AckNack();
172
173
174
175             ackNack.ReaderId = readerId;
176             ackNack.WriterId = writerId;
177             ackNack.ReaderSNState = sns ;
178
179             ackNack.Count =  countaux;
180             msg.SubMessages.Add(ackNack);
181
182             SendData(msg);
183
184
185         }
186         */
187         /// <summary>
188         /// Writes a message to network
189         /// </summary>
190         /// <param name="msg"></param>
191         /// <returns></returns>
192         private void SendData(Message msg)
193         {
194             trans.SendMessage(msg);
195         }
196     }
197 }
198
```

```
1  using Doopec.Rtps.Messages;
2  using Doopec.Serializer;
3  using Doopec.Utils.Transport;
4  using log4net;
5  using Mina.Core.Buffer;
6  using Rtps.Behavior;
7  using Rtps.Messages;
8  using Rtps.Messages.Types;
9  using Rtps.Structure;
10 using Rtps.Structure.Types;
11 using System;
12 using System.Collections.Generic;
13 using System.Linq;
14 using System.Reflection;
15 using System.Text;
16 using Data = Rtps.Messages.Submessages.Data;
17 using DataObj = Rtps.Structure.Types.Data;
18
19 namespace Doopec.Rtps.Behavior
20 {
21     public class RtpsStatelessReader<T> : StatelessReader<T>, IDisposable where T : new()
22     {
23         protected static readonly ILog log = LogManager.GetLogger(MethodBase.GetCurrentMethod().
       DeclaringType);
24         protected List<UDPReceiver> UDPReceivers { get; private set; }
25
26         public RtpsStatelessReader(GUID guid)
27             : base(guid)
28         {
29             Doopec.Serializer.Serializer.Initialize(typeof(T));
30             UDPReceivers = new List<UDPReceiver>();
31         }
32
33         protected void InitReceivers()
34         {
35             foreach (var locator in MulticastLocatorList)
36             {
37                 UDPReceiver rec = new UDPReceiver(locator, 1024);
38                 rec.ParticipantId = this.Guid;
39                 rec.MessageReceived += NewMessage;
40                 UDPReceivers.Add(rec);
41             }
42
43             foreach (var locator in UnicastLocatorList)
44             {
45                 UDPReceiver rec = new UDPReceiver(locator, 1024);
46                 rec.ParticipantId = this.Guid;
47                 rec.MessageReceived += NewMessage;
48                 UDPReceivers.Add(rec);
49             }
50         }
51
52         protected void StartReceivers()
53         {
54             foreach (var rec in UDPReceivers)
55             {
56                 rec.Start();
57             }
58         }
59
60         protected virtual void NewMessage(object sender, RTPSMessageEventArgs e)
61         {
62             Message msg = e.Message;
63             log.DebugFormat("New Message has arrived from {0}", e.Session.RemoteEndPoint);
64             log.DebugFormat("Message Header: {0}", msg.Header);
65             foreach (var submsg in msg.SubMessages)
66             {
67                 switch (submsg.Kind)
68                 {
69                     case SubMessageKind.DATA:
70                         Data d = submsg as Data;
71                         log.DebugFormat("SubMessage Data: {0}", submsg.Kind);
72                         log.DebugFormat("The KeyFlag value state is: {0}", d.HasKeyFlag);
73                         log.DebugFormat("The DataFlag value state is: {0}", d.HasDataFlag);
```

```
74                          log.DebugFormat("The InlineQoSFlag value state is: {0}", d.HasInlineQosFlag);
75                          log.DebugFormat("The EndiannessFlag value state is: {0}", d.Header.Flags.
    IsLittleEndian);
76                          log.DebugFormat("The octetsToNextHeader value is: {0}", d.Header.
    SubMessageLength);
77                          log.DebugFormat("The extraFlags value is: {0}", d.ExtraFlags.Value);
78                          log.DebugFormat("The octetsToInlineQos value is: Aun no logro");
79                          log.DebugFormat("The readerID is: {0}", d.ReaderId);
80                          log.DebugFormat("The writerID is: {0}", d.WriterId);
81                          log.DebugFormat("The writerSN is: {0}", d.WriterSN);
82
83                          IoBuffer buf = IoBuffer.Wrap(d.SerializedPayload.DataEncapsulation.
    SerializedPayload);
84                          buf.Order = (d.Header.IsLittleEndian ? ByteOrder.LittleEndian : ByteOrder.
    BigEndian);
85                          T obj = EncapsulationManager.Deserialize<T>(buf);
86  #if TODO
87                          CacheChange<T> change = new CacheChange<T>(ChangeKind.ALIVE, new GUID(msg.Header.
    GuidPrefix, d.WriterId), d.WriterSN, new DataObj(obj), new InstanceHandle());
88                          ReaderCache.AddChange(change);
89  #endif
90                          break;
91                      default:
92                          log.DebugFormat("SubMessage: {0}", submsg.Kind);
93                          break;
94                  }
95              }
96          }
97
98          public void Dispose()
99          {
100             foreach (var rec in UDPReceivers)
101             {
102                 rec.MessageReceived -= NewMessage;
103                 rec.Close();
104                 rec.Dispose();
105             }
106             UDPReceivers.Clear();
107         }
108     }
109 }
110
```

```
1  using Doopec.Serializer;
2  using Doopec.Utils.Transport;
3  using Doopec.Encoders;
4  using log4net;
5  using Mina.Core.Buffer;
6  using Rtps.Behavior;
7  using Rtps.Messages;
8  using Rtps.Messages.Submessages;
9  using Rtps.Messages.Submessages.Elements;
10 using Rtps.Structure;
11 using Rtps.Structure.Types;
12 using System;
13 using System.Collections.Generic;
14 using System.Linq;
15 using System.Reflection;
16 using System.Text;
17 using Data = Rtps.Messages.Submessages.Data;
18 using Doopec.Rtps.Messages;
19 using Doopec.Serializer.Attributes;
20
21 namespace Doopec.Rtps.Behavior
22 {
23     public class RtpsStatelessWriter<T> : StatelessWriter<T>, IDisposable where T : new()
24     {
25         protected static readonly ILog log = LogManager.GetLogger(MethodBase.GetCurrentMethod().
       DeclaringType);
26         protected List<UDPTransmitter> UDPTransmitters { get; private set; }
27         private WriterWorker worker;
28         protected Encapsulation Scheme { get; set; }
29
30         public RtpsStatelessWriter(GUID guid)
31             : base(guid)
32         {
33             Doopec.Serializer.Serializer.Initialize(typeof(T));
34             UDPTransmitters = new List<UDPTransmitter>();
35             Scheme = Encapsulation.CDR_BE;
36         }
37
38         protected void InitTransmitters()
39         {
40             foreach (var locator in MulticastLocatorList)
41             {
42                 UDPTransmitter rec = new UDPTransmitter(locator, 1024);
43                 rec.ParticipantId = this.Guid;
44                 UDPTransmitters.Add(rec);
45             }
46
47             // TODO. Just for testing. I dont like so many messages
48             //foreach (var locator in UnicastLocatorList)
49             //{
50             //    UDPTransmitter trans = new UDPTransmitter(locator, 1024);
51             //    trans.ParticipantId = this.Guid;
52             //    UDPTransmitters.Add(trans);
53             //}
54             worker = new WriterWorker(this.PeriodicWork);
55         }
56
57         protected void StartTransmitters()
58         {
59             foreach (var trans in UDPTransmitters)
60             {
61                 trans.Start();
62             }
63             worker.Start((int)this.ResendDataPeriod.AsMillis());
64         }
65         protected virtual void PeriodicWork()
66         {
67             // the RTPS Writer to repeatedly announce the availability of data by sending a Heartbeat
       Message.
68             log.DebugFormat("I have to send a Heartbeat Message,  at {0}", DateTime.Now);
69             SendHeartbeat();
70             if (HistoryCache.Changes.Count > 0)
71             {
72                 foreach (var change in HistoryCache.Changes)
```

```
73                {
74                    //SendHeartbeat();
75                    //SendData(change);
76                    SendDataHeartbeat(change);
77                }
78                HistoryCache.Changes.Clear(); //TODO
79            }
80        }
81        private void SendHeartbeat()
82        {
83            // Create a Message with Heartbeat
84            Message m1 = new Message();
85
86            Heartbeat heartbeat = new Heartbeat();
87            EntityId id1 = EntityId.ENTITYID_UNKNOWN;
88            EntityId id2 = EntityId.ENTITYID_PARTICIPANT;
89
90            heartbeat.readerId = id1;
91            heartbeat.writerId = id2;
92            heartbeat.firstSN = new SequenceNumber(10);
93            heartbeat.lastSN = new SequenceNumber(20);
94            heartbeat.count = 5;
95            m1.SubMessages.Add(heartbeat);
96
97            SendData(m1);
98        }
99
100       public void SendData(CacheChange<T> change)
101       {
102           // Create a Message with InfoSource
103           Message msg = new Message();
104           EntityId readerId = EntityId.ENTITYID_UNKNOWN;
105           EntityId writerId = change.WriterGuid.EntityId;
106           SerializedPayload payload = new SerializedPayload();
107           IoBuffer buff = IoBuffer.Allocate(1024);
108
109           payload.DataEncapsulation = EncapsulationManager.Serialize<T>((T)change.DataValue.Value,
    Scheme);
110           Data data = new Data(readerId, writerId, change.SequenceNumber.LongValue, null, payload);
111           msg.SubMessages.Add(data);
112
113           // Write Message to bytes1 array
114           SendData(msg);
115       }
116
117       public void SendDataHeartbeat(CacheChange<T> change)
118       {
119           // Create a Message with InfoSource
120           Message msg = new Message();
121           EntityId readerId = EntityId.ENTITYID_UNKNOWN;
122           EntityId writerId = change.WriterGuid.EntityId;
123           SerializedPayload payload = new SerializedPayload();
124           IoBuffer buff = IoBuffer.Allocate(1024);
125           payload.DataEncapsulation = buff.EncapsuleCDRData(change.DataValue.Value, BitConverter.
    IsLittleEndian ? ByteOrder.LittleEndian : ByteOrder.BigEndian);
126           Data data = new Data(readerId, writerId, change.SequenceNumber.LongValue, null, payload);
127           msg.SubMessages.Add(data);
128
129           Heartbeat heartbeat = new Heartbeat();
130           heartbeat.readerId = readerId;
131           heartbeat.writerId = writerId;
132           heartbeat.firstSN = change.SequenceNumber;
133           heartbeat.lastSN = change.SequenceNumber;
134           heartbeat.count = 1;
135           msg.SubMessages.Add(heartbeat);
136
137           SendData(msg);
138       }
139
140       /// <summary>
141       /// Writes a message to network
142       /// </summary>
143       /// <param name="msg"></param>
144       /// <returns></returns>
```

```
145         private void SendData(Message msg)
146         {
147             foreach (var trans in UDPTransmitters)
148                 trans.SendMessage(msg);
149         }
150 
151         public void Dispose()
152         {
153             worker.End();
154             foreach (var trans in UDPTransmitters)
155             {
156                 trans.Close();
157                 trans.Dispose();
158             }
159             UDPTransmitters.Clear();
160         }
161     }
162 }
163 
```

```
using log4net;
using Doopec.Rtps.Utils;
using Doopec.Utils.Transport;
using Mina.Core.Buffer;
using Rtps.Messages;
using Rtps.Messages.Submessages;
using Rtps.Messages.Submessages.Elements;
using Rtps.Structure;
using Rtps.Structure.Types;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Reflection;
using System.Text;
using System.Threading.Tasks;


namespace Doopec.Rtps.Behavior
{
    public class WriterWorker : PeriodicWorker
    {
        private static readonly ILog log = LogManager.GetLogger(MethodBase.GetCurrentMethod().
    DeclaringType);

        public delegate void PeriodicWorkDelegate();

        private PeriodicWorkDelegate periodicWork;

        public WriterWorker( )
        {

        }
        public override void End()
        {

        }

        public WriterWorker(PeriodicWorkDelegate periodicWork)
        {
            this.periodicWork = periodicWork;
        }

        public override void DoPeriodicWork()
        {
            base.DoPeriodicWork();
            if (periodicWork != null)
                periodicWork();
        }


    }
}

```

```
1  using org.omg.dds.topic;
2  using System;
3  using System.Collections.Generic;
4  using System.Linq;
5  using System.Text;
6  using System.Threading.Tasks;
7
8  namespace Doopec.Rtps.Discovery
9  {
10     public class DiscoveredReaderData : SubscriptionBuiltinTopicData
11     {
12         public override BuiltinTopicKey Key
13         {
14             get
15             {
16                 throw new NotImplementedException();
17             }
18         }
19
20         public override BuiltinTopicKey ParticipantKey
21         {
22             get
23             {
24                 throw new NotImplementedException();
25             }
26         }
27
28         public override string TopicName
29         {
30             get
31             {
32                 throw new NotImplementedException();
33             }
34         }
35
36         public override string TypeName
37         {
38             get
39             {
40                 throw new NotImplementedException();
41             }
42         }
43
44         public override List<string> EquivalentTypeName
45         {
46             get
47             {
48                 throw new NotImplementedException();
49             }
50         }
51
52         public override List<string> BaseTypeName
53         {
54             get
55             {
56                 throw new NotImplementedException();
57             }
58         }
59
60         public override org.omg.dds.type.typeobject.TypeObject Type
61         {
62             get
63             {
64                 throw new NotImplementedException();
65             }
66         }
67
68         public override org.omg.dds.core.policy.DurabilityQosPolicy Durability
69         {
70             get
71             {
72                 throw new NotImplementedException();
73             }
74         }
```

```
75
76          public override org.omg.dds.core.policy.DeadlineQosPolicy Deadline
77          {
78              get
79              {
80                  throw new NotImplementedException();
81              }
82          }
83
84          public override org.omg.dds.core.policy.LatencyBudgetQosPolicy LatencyBudget
85          {
86              get
87              {
88                  throw new NotImplementedException();
89              }
90          }
91
92          public override org.omg.dds.core.policy.LivelinessQosPolicy Liveliness
93          {
94              get
95              {
96                  throw new NotImplementedException();
97              }
98          }
99
100         public override org.omg.dds.core.policy.ReliabilityQosPolicy Reliability
101         {
102             get
103             {
104                 throw new NotImplementedException();
105             }
106         }
107
108         public override org.omg.dds.core.policy.OwnershipQosPolicy Ownership
109         {
110             get
111             {
112                 throw new NotImplementedException();
113             }
114         }
115
116         public override org.omg.dds.core.policy.DestinationOrderQosPolicy DestinationOrder
117         {
118             get
119             {
120                 throw new NotImplementedException();
121             }
122         }
123
124         public override org.omg.dds.core.policy.UserDataQosPolicy UserData
125         {
126             get
127             {
128                 throw new NotImplementedException();
129             }
130         }
131
132         public override org.omg.dds.core.policy.TimeBasedFilterQosPolicy TimeBasedFilter
133         {
134             get
135             {
136                 throw new NotImplementedException();
137             }
138         }
139
140         public override org.omg.dds.core.policy.PresentationQosPolicy Presentation
141         {
142             get
143             {
144                 throw new NotImplementedException();
145             }
146         }
147
148         public override org.omg.dds.core.policy.PartitionQosPolicy Partition
```

```
149         {
150             get
151             {
152                 throw new NotImplementedException();
153             }
154         }
155 
156         public override org.omg.dds.core.policy.TopicDataQosPolicy TopicData
157         {
158             get
159             {
160                 throw new NotImplementedException();
161             }
162         }
163 
164         public override org.omg.dds.core.policy.GroupDataQosPolicy GroupData
165         {
166             get
167             {
168                 throw new NotImplementedException();
169             }
170         }
171 
172         public override org.omg.dds.core.policy.DataRepresentationQosPolicy Representation
173         {
174             get
175             {
176                 throw new NotImplementedException();
177             }
178         }
179 
180         public override org.omg.dds.core.policy.TypeConsistencyEnforcementQosPolicy TypeConsistency
181         {
182             get
183             {
184                 throw new NotImplementedException();
185             }
186         }
187 
188         public override SubscriptionBuiltinTopicData CopyFrom(SubscriptionBuiltinTopicData other)
189         {
190             throw new NotImplementedException();
191         }
192 
193         public override SubscriptionBuiltinTopicData FinishModification()
194         {
195             throw new NotImplementedException();
196         }
197 
198         public override SubscriptionBuiltinTopicData Modify()
199         {
200             throw new NotImplementedException();
201         }
202 
203         public override org.omg.dds.core.Bootstrap GetBootstrap()
204         {
205             throw new NotImplementedException();
206         }
207     }
208 }
209 
```

```
using org.omg.dds.topic;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Doopec.Rtps.Discovery
{
    public class DiscoveredTopicData : TopicBuiltinTopicData
    {
        public override BuiltinTopicKey Key
        {
            get
            {
                throw new NotImplementedException();
            }
        }

        public override string Name
        {
            get
            {
                throw new NotImplementedException();
            }
        }

        public override string TypeName
        {
            get
            {
                throw new NotImplementedException();
            }
        }

        public override List<string> EquivalentTypeName
        {
            get
            {
                throw new NotImplementedException();
            }
        }

        public override List<string> BaseTypeName
        {
            get
            {
                throw new NotImplementedException();
            }
        }

        public override org.omg.dds.type.typeobject.TypeObject Type
        {
            get
            {
                throw new NotImplementedException();
            }
        }

        public override org.omg.dds.core.policy.DurabilityQosPolicy Durability
        {
            get
            {
                throw new NotImplementedException();
            }
        }

        public override org.omg.dds.core.policy.DurabilityServiceQosPolicy DurabilityService
        {
            get
            {
                throw new NotImplementedException();
            }
        }
```

```
75
76          public override org.omg.dds.core.policy.DeadlineQosPolicy Deadline
77          {
78              get
79              {
80                  throw new NotImplementedException();
81              }
82          }
83
84          public override org.omg.dds.core.policy.LatencyBudgetQosPolicy LatencyBudget
85          {
86              get
87              {
88                  throw new NotImplementedException();
89              }
90          }
91
92          public override org.omg.dds.core.policy.LivelinessQosPolicy Liveliness
93          {
94              get
95              {
96                  throw new NotImplementedException();
97              }
98          }
99
100         public override org.omg.dds.core.policy.ReliabilityQosPolicy Reliability
101         {
102             get
103             {
104                 throw new NotImplementedException();
105             }
106         }
107
108         public override org.omg.dds.core.policy.TransportPriorityQosPolicy TransportPriority
109         {
110             get
111             {
112                 throw new NotImplementedException();
113             }
114         }
115
116         public override org.omg.dds.core.policy.LifespanQosPolicy Lifespan
117         {
118             get
119             {
120                 throw new NotImplementedException();
121             }
122         }
123
124         public override org.omg.dds.core.policy.DestinationOrderQosPolicy DestinationOrder
125         {
126             get
127             {
128                 throw new NotImplementedException();
129             }
130         }
131
132         public override org.omg.dds.core.policy.HistoryQosPolicy History
133         {
134             get
135             {
136                 throw new NotImplementedException();
137             }
138         }
139
140         public override org.omg.dds.core.policy.ResourceLimitsQosPolicy ResourceLimits
141         {
142             get
143             {
144                 throw new NotImplementedException();
145             }
146         }
147
148         public override org.omg.dds.core.policy.OwnershipQosPolicy Ownership
```

```
149         {
150             get
151             {
152                 throw new NotImplementedException();
153             }
154         }
155 
156         public override org.omg.dds.core.policy.TopicDataQosPolicy TopicData
157         {
158             get
159             {
160                 throw new NotImplementedException();
161             }
162         }
163 
164         public override org.omg.dds.core.policy.DataRepresentationQosPolicy Representation
165         {
166             get
167             {
168                 throw new NotImplementedException();
169             }
170         }
171 
172         public override org.omg.dds.core.policy.TypeConsistencyEnforcementQosPolicy TypeConsistency
173         {
174             get
175             {
176                 throw new NotImplementedException();
177             }
178         }
179 
180         public override TopicBuiltinTopicData CopyFrom(TopicBuiltinTopicData other)
181         {
182             throw new NotImplementedException();
183         }
184 
185         public override TopicBuiltinTopicData FinishModification()
186         {
187             throw new NotImplementedException();
188         }
189 
190         public override TopicBuiltinTopicData Modify()
191         {
192             throw new NotImplementedException();
193         }
194 
195         public override org.omg.dds.core.Bootstrap GetBootstrap()
196         {
197             throw new NotImplementedException();
198         }
199     }
200 }
201 
```

```
1  using org.omg.dds.topic;
2
3  namespace Doopec.Rtps.Discovery
4  {
5      public class DiscoveredWriterData : PublicationBuiltinTopicData
6      {
7          public override BuiltinTopicKey Key
8          {
9              get
10             {
11                 throw new System.NotImplementedException();
12             }
13         }
14
15         public override BuiltinTopicKey ParticipantKey
16         {
17             get
18             {
19                 throw new System.NotImplementedException();
20             }
21         }
22
23         public override string TopicName
24         {
25             get
26             {
27                 throw new System.NotImplementedException();
28             }
29         }
30
31         public override string TypeName
32         {
33             get
34             {
35                 throw new System.NotImplementedException();
36             }
37         }
38
39         public override System.Collections.Generic.List<string> EquivalentTypeName
40         {
41             get
42             {
43                 throw new System.NotImplementedException();
44             }
45         }
46
47         public override System.Collections.Generic.List<string> BaseTypeName
48         {
49             get
50             {
51                 throw new System.NotImplementedException();
52             }
53         }
54
55         public override org.omg.dds.type.typeobject.TypeObject Type
56         {
57             get
58             {
59                 throw new System.NotImplementedException();
60             }
61         }
62
63         public override org.omg.dds.core.policy.DurabilityQosPolicy Durability
64         {
65             get
66             {
67                 throw new System.NotImplementedException();
68             }
69         }
70
71         public override org.omg.dds.core.policy.DurabilityServiceQosPolicy DurabilityService
72         {
73             get
74             {
```

```
75                  throw new System.NotImplementedException();
76              }
77          }
78
79          public override org.omg.dds.core.policy.DeadlineQosPolicy Deadline
80          {
81              get
82              {
83                  throw new System.NotImplementedException();
84              }
85          }
86
87          public override org.omg.dds.core.policy.LatencyBudgetQosPolicy LatencyBudget
88          {
89              get
90              {
91                  throw new System.NotImplementedException();
92              }
93          }
94
95          public override org.omg.dds.core.policy.LivelinessQosPolicy Liveliness
96          {
97              get
98              {
99                  throw new System.NotImplementedException();
100             }
101         }
102
103         public override org.omg.dds.core.policy.ReliabilityQosPolicy Reliability
104         {
105             get
106             {
107                 throw new System.NotImplementedException();
108             }
109         }
110
111         public override org.omg.dds.core.policy.LifespanQosPolicy Lifespan
112         {
113             get
114             {
115                 throw new System.NotImplementedException();
116             }
117         }
118
119         public override org.omg.dds.core.policy.UserDataQosPolicy UserData
120         {
121             get
122             {
123                 throw new System.NotImplementedException();
124             }
125         }
126
127         public override org.omg.dds.core.policy.OwnershipQosPolicy Ownership
128         {
129             get
130             {
131                 throw new System.NotImplementedException();
132             }
133         }
134
135         public override org.omg.dds.core.policy.OwnershipStrengthQosPolicy OwnershipStrength
136         {
137             get
138             {
139                 throw new System.NotImplementedException();
140             }
141         }
142
143         public override org.omg.dds.core.policy.DestinationOrderQosPolicy DestinationOrder
144         {
145             get
146             {
147                 throw new System.NotImplementedException();
148             }
```

```
149         }
150
151         public override org.omg.dds.core.policy.PresentationQosPolicy Presentation
152         {
153             get
154             {
155                 throw new System.NotImplementedException();
156             }
157         }
158
159         public override org.omg.dds.core.policy.PartitionQosPolicy Partition
160         {
161             get
162             {
163                 throw new System.NotImplementedException();
164             }
165         }
166
167         public override org.omg.dds.core.policy.TopicDataQosPolicy TopicData
168         {
169             get
170             {
171                 throw new System.NotImplementedException();
172             }
173         }
174
175         public override org.omg.dds.core.policy.GroupDataQosPolicy GroupData
176         {
177             get
178             {
179                 throw new System.NotImplementedException();
180             }
181         }
182
183         public override org.omg.dds.core.policy.DataRepresentationQosPolicy Representation
184         {
185             get
186             {
187                 throw new System.NotImplementedException();
188             }
189         }
190
191         public override org.omg.dds.core.policy.TypeConsistencyEnforcementQosPolicy TypeConsistency
192         {
193             get
194             {
195                 throw new System.NotImplementedException();
196             }
197         }
198
199         public override PublicationBuiltinTopicData CopyFrom(PublicationBuiltinTopicData other)
200         {
201             throw new System.NotImplementedException();
202         }
203
204         public override PublicationBuiltinTopicData FinishModification()
205         {
206             throw new System.NotImplementedException();
207         }
208
209         public override PublicationBuiltinTopicData Clone()
210         {
211             throw new System.NotImplementedException();
212         }
213
214         public override PublicationBuiltinTopicData Modify()
215         {
216             throw new System.NotImplementedException();
217         }
218
219         public override org.omg.dds.core.Bootstrap GetBootstrap()
220         {
221             throw new System.NotImplementedException();
222         }
```

```
223     }
224 }
225
```

```
using Doopec.Rtps.Utils;
using org.omg.dds.domain;
using Rtps.Structure.Types;
using System;
using System.Collections.Generic;

namespace Doopec.Rtps.Discovery
{
    /// <summary>
    /// Discovery Strategy class that implements RTPS IsDiscovery
    /// This class implements the Discovery interface for Rtps-based
    /// IsDiscovery.
    /// </summary>
    public class DiscoveryImpl : IDisposable
    {
        GuidGenerator generator = new GuidGenerator();
        public void StartDiscovery()
        {
        }

        public void CloseDiscovery()
        {
        }

        // Participant operations:
        public virtual bool AttachParticipant(int domainId, int participantId)
        {
            throw new NotImplementedException();
        }

        internal virtual AddDomainStatus AddDomainParticipant(int domain, DomainParticipantQos qos)
        {
            lock (this)
            {
                AddDomainStatus ads = new AddDomainStatus() { id = new GUID(), federated = false };
                generator.Populate(ref ads.id);
                ads.id.EntityId = EntityId.ENTITYID_PARTICIPANT;
                try
                {
                    if (participants_.ContainsKey(domain) && participants_[domain] != null)
                    {
                        participants_[domain][ads.id] = new Spdp(domain, ads.id, qos, this);
                    }
                    else
                    {
                        participants_[domain] = new Dictionary<GUID, Spdp>();
                        participants_[domain][ads.id] = new Spdp(domain, ads.id, qos, this);
                    }
                }
                catch (Exception e)
                {
                    ads.id = GUID.GUID_UNKNOWN;
                    //  ACE_ERROR((LM_ERROR, "(%P|%t) RtpsDiscovery::add_domain_participant() - "
                    //    "failed to initialize RTPS Simple Participant Discovery Protocol: %C\n",
                    //    e.what()));
                }
                return ads;
            }
        }


        public virtual bool RemoveDomainParticipant(int domainId, int participantId)
        {
            throw new NotImplementedException();
        }


        public virtual bool IgnoreDomainParticipant(int domainId, int myParticipantId, int ignoreId)
        {
            throw new NotImplementedException();
        }


        public virtual bool UpdateDomainParticipantQos(int domain, int articipantId, DomainParticipantQos ↙
```

```
    qos)
        {
            throw new NotImplementedException();
        }

        private IDictionary<int, IDictionary<GUID, Spdp>> participants_ = new Dictionary<int, IDictionary ↙
    <GUID, Spdp>>();


        public void Dispose()
        {
            this.CloseDiscovery();
        }
    }

    // Information returned from call to add_domain_participant()
    internal class AddDomainStatus
    {
        // These are unique across a domain
        // They are also the InstanceHandle_t in Sample_Info for built-in Topics
        public GUID id;
        public bool federated;
    }
}

```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace Doopec.Rtps.Discovery
8  {
9      public class Sedp
10     {
11     }
12 }
13
```

```
1  using Rtps.Behavior;
2  using Rtps.Structure;
3  using Rtps.Structure.Types;
4
5  namespace Doopec.Rtps.Discovery
6  {
7      public class SEDPbuiltinPublicationsReader : StatefulReader<DiscoveredWriterData>
8      {
9          public SEDPbuiltinPublicationsReader(GUID guid)
10             : base(guid)
11         {
12             this.guid = new GUID(guid.Prefix, EntityId.ENTITYID_SEDP_BUILTIN_PUBLICATIONS_READER);
13         }
14     }
15 }
16
```

```
1  using Rtps.Behavior;
2  using Rtps.Structure;
3  using Rtps.Structure.Types;
4
5  namespace Doopec.Rtps.Discovery
6  {
7      public class SEDPbuiltinPublicationsWriter : StatefulWriter<DiscoveredWriterData>
8      {
9          public SEDPbuiltinPublicationsWriter(GUID guid)
10             : base(guid)
11         {
12             this.guid = new GUID(guid.Prefix, EntityId.ENTITYID_SEDP_BUILTIN_PUBLICATIONS_WRITER);
13         }
14
15     }
16 }
17
```

```
1  using Rtps.Behavior;
2  using Rtps.Structure;
3  using Rtps.Structure.Types;
4
5  namespace Doopec.Rtps.Discovery
6  {
7      public class SEDPbuiltinSubscriptionsReader : StatefulReader<DiscoveredReaderData>
8      {
9          public SEDPbuiltinSubscriptionsReader(GUID guid)
10             : base(guid)
11         {
12             this.guid = new GUID(guid.Prefix, EntityId.ENTITYID_SEDP_BUILTIN_SUBSCRIPTIONS_READER);
13         }
14
15     }
16 }
17
```

```
1  using Rtps.Behavior;
2  using Rtps.Structure;
3  using Rtps.Structure.Types;
4
5  namespace Doopec.Rtps.Discovery
6  {
7      public class SEDPbuiltinSubscriptionsWriter : StatefulWriter<DiscoveredReaderData>
8      {
9          public SEDPbuiltinSubscriptionsWriter(GUID guid)
10             : base(guid)
11         {
12             this.guid = new GUID(guid.Prefix, EntityId.ENTITYID_SEDP_BUILTIN_SUBSCRIPTIONS_WRITER);
13         }
14
15     }
16 }
17
```

```
1  using Rtps.Behavior;
2  using Rtps.Structure;
3  using Rtps.Structure.Types;
4
5  namespace Doopec.Rtps.Discovery
6  {
7      public class SEDPbuiltinTopicsReader : StatefulReader<DiscoveredTopicData>
8      {
9          public SEDPbuiltinTopicsReader(GUID guid)
10             : base(guid)
11         {
12             this.guid = new GUID(guid.Prefix, EntityId.ENTITYID_SEDP_BUILTIN_TOPIC_READER);
13         }
14
15     }
16 }
17
```

```
1  using Rtps.Behavior;
2  using Rtps.Structure;
3  using Rtps.Structure.Types;
4
5  namespace Doopec.Rtps.Discovery
6  {
7      public class SEDPbuiltinTopicsWriter : StatefulWriter<DiscoveredTopicData>
8      {
9          public SEDPbuiltinTopicsWriter(GUID guid)
10             : base(guid)
11         {
12             this.guid = new GUID(guid.Prefix, EntityId.ENTITYID_SEDP_BUILTIN_TOPIC_WRITER);
13         }
14
15     }
16 }
17
```

```
1  using org.omg.dds.domain;
2  using Rtps.Structure.Types;
3  using System;
4  using System.Collections.Generic;
5  using System.Linq;
6  using System.Text;
7  using System.Threading.Tasks;
8
9  namespace Doopec.Rtps.Discovery
10 {
11     /// <summary>
12     /// Each instance of class Spdp represents the implementation of the RTPS
13     /// Simple Participant Discovery Protocol for a single local DomainParticipant.
14     /// </summary>
15     public class Spdp
16     {
17         public Spdp(int domain, GUID id, DomainParticipantQos qos, DiscoveryImpl disco)
18         { }
19     }
20 }
21
```

```
1  using Doopec.Configuration;
2  using Doopec.Configuration.Rtps;
3  using Doopec.Rtps.Behavior;
4  using Doopec.Rtps.Structure;
5  using Rtps.Behavior.Types;
6  using Rtps.Discovery.Spdp;
7  using Rtps.Structure;
8  using Rtps.Structure.Types;
9  using System;
10 using System.Collections.Generic;
11 using System.Net;
12
13 namespace Doopec.Rtps.Discovery
14 {
15     public class SPDPbuiltinParticipantReaderImpl : RtpsStatelessReader<SPDPdiscoveredParticipantData>
16     {
17         public SPDPbuiltinParticipantReaderImpl(Transport transportconfig, ParticipantImpl participant)
18             : base(participant.Guid)
19         {
20             this.TopicKind = TopicKind.WITH_KEY;
21             this.ReliabilityLevel = ReliabilityKind.BEST_EFFORT;
22             this.ExpectsInlineQos = false;
23             //The following timing-related values are used as the defaults in order to facilitate
24             // ‘out-of-the-box’ interoperability between implementations.
25             this.HeartbeatResponseDelay = new Duration(transportconfig.RtpsReader.HeartbeatResponseDelay.↙
       Val); // default is 500 milliseconds
26             this.HeartbeatSuppressionDuration = new Duration(transportconfig.RtpsReader.                ↙
       HeartbeatSuppressionDuration.Val);// default is 0 milliseconds
27
28
29             SetLocatorListFromConfig(transportconfig, participant);
30             InitReceivers();
31             foreach (var rec in this.UDPReceivers)
32             {
33                 rec.IsDiscovery = true;
34             }
35         }
36
37         public void Start()
38         {
39             StartReceivers();
40         }
41
42         protected void SetLocatorListFromConfig(Transport transport, ParticipantImpl participant)
43         {
44             IList<Locator> unicastLocatorList = new List<Locator>();
45             IList<Locator> multicastLocatorList = new List<Locator>();
46
47             int PB = transport.Discovery.PortBase.Val;
48             int DG = transport.Discovery.DomainGain.Val;
49             int PG = transport.Discovery.ParticipantGain.Val;
50             int d0 = transport.Discovery.OffsetMetatrafficMulticast.Val;
51             int d1 = transport.Discovery.OffsetMetatrafficUnicast.Val;
52
53             string[] unicastStr = transport.Discovery.MetatrafficUnicastLocatorList.Val.Split(new char[] ↙
       { ',', ';' }, StringSplitOptions.RemoveEmptyEntries);
54             foreach (string addr in unicastStr)
55             {
56                 string[] parts = addr.Split(':');
57                 IPAddress[] addresses = System.Net.Dns.GetHostAddresses(parts[0]);
58                 IPAddress ipaddr = null;
59                 if (addresses != null)
60                 {
61                     foreach (var ipa in addresses)
62                         if (ipa.AddressFamily == System.Net.Sockets.AddressFamily.InterNetwork)
63                         {
64                             ipaddr = ipa;
65                             break;
66                         }
67                 }
68                 else
69                     throw new ArgumentException("Invalid unicast address " + parts[0]);
70
71                 int port;
```

```
72                  if (parts.Length <= 1)
73                  {
74                      port = PB + DG * participant.DomainId + d1 + PG * participant.ParticipantId;
75                  }
76                  else
77                      port = int.Parse(parts[1]);
78
79                  if (ipaddr != null)
80                  {
81                      Locator locator = new Locator(ipaddr, port);
82                      log.DebugFormat("Using unicast Addr:{0} and Port:{0} for SPDP Discovery", ipaddr,
      port);
83                      unicastLocatorList.Add(locator);
84                  }
85              }
86
87              string[] multicastStr = transport.Discovery.MetatrafficMulticastLocatorList.Val.Split(new
      char[] { ',', ';' }, StringSplitOptions.RemoveEmptyEntries);
88              foreach (string addr in multicastStr)
89              {
90                  string[] parts = addr.Split(':');
91                  IPAddress[] addresses = System.Net.Dns.GetHostAddresses(parts[0]);
92                  IPAddress ipaddr = null;
93                  if (addresses != null)
94                  {
95                      foreach (var ipa in addresses)
96                          if (ipa.AddressFamily == System.Net.Sockets.AddressFamily.InterNetwork)
97                          {
98                              ipaddr = ipa;
99                              break;
100                         }
101                 }
102                 else
103                     throw new ArgumentException("Invalid multicast address " + parts[0]);
104
105                 int port;
106                 if (parts.Length <= 1)
107                 {
108                     port = PB + DG * participant.DomainId + d0;
109                 }
110                 else
111                     port = int.Parse(parts[1]);
112                 if (ipaddr != null)
113                 {
114                     Locator locator = new Locator(ipaddr, port);
115                     log.DebugFormat("Using multicast Addr:{0} and Port:{0} for SPDP Discovery", ipaddr,
      port);
116                     multicastLocatorList.Add(locator);
117                 }
118             }
119
120             this.UnicastLocatorList = unicastLocatorList;
121             this.MulticastLocatorList = multicastLocatorList;
122         }
123     }
124 }
125
```

```
1  using Doopec.Configuration.Rtps;
2  using Doopec.Rtps.Behavior;
3  using Doopec.Rtps.Structure;
4  using Doopec.Serializer.Attributes;
5  using Rtps.Behavior;
6  using Rtps.Behavior.Types;
7  using Rtps.Discovery.Sedp;
8  using Rtps.Discovery.Spdp;
9  using Rtps.Structure;
10 using Rtps.Structure.Types;
11 using System;
12 using System.Collections.Generic;
13 using System.Net;
14
15 namespace Doopec.Rtps.Discovery
16 {
17     /// <summary>
18     /// For each Participant, the SPDP creates two RTPS built-in Endpoints: the
19     /// SPDPbuiltinParticipantWriter and the SPDPbuiltinParticipantReader.
20     /// The SPDPbuiltinParticipantWriter is an RTPS Best-Effort StatelessWriter. The HistoryCache of the
21     /// SPDPbuiltinParticipantWriter contains a single data-object of type SPDPdiscoveredParticipantData.
22     /// The Value of this data-object is Set from the attributes in the Participant.
23     /// If the attributes change, the data-object is replaced.
24     /// </summary>
25     public class SPDPbuiltinParticipantWriterImpl : RtpsStatelessWriter<SPDPdiscoveredParticipantData>
26     {
27         private WriterWorker worker;
28
29         public SPDPbuiltinParticipantWriterImpl(Transport transportconfig, ParticipantImpl participant)
30             : base(participant.Guid)
31         {
32             SetLocatorListFromConfig(transportconfig, participant);
33             participant.DefaultMulticastLocatorList = this.MulticastLocatorList as List<Locator>;
34             participant.DefaultUnicastLocatorList = this.UnicastLocatorList as List<Locator>;
35             SPDPdiscoveredParticipantData data = new SPDPdiscoveredParticipantData(participant);
36             // TODO Assign UserData from configuration
37             CacheChange<SPDPdiscoveredParticipantData> change = this.NewChange(ChangeKind.ALIVE, new Data
       (data), null);
38             this.HistoryCache.AddChange(change);
39
40             this.TopicKind = TopicKind.WITH_KEY;
41             this.ReliabilityLevel = ReliabilityKind.BEST_EFFORT;
42             this.ResendDataPeriod = new Duration(transportconfig.Discovery.ResendPeriod.Val);
43             this.heartbeatPeriod = new Duration(transportconfig.RtpsWriter.HeartbeatPeriod.Val);
44
45             //The following timing-related values are used as the defaults in order to facilitate
46             // 'out-of-the-box' interoperability between implementations.
47             this.nackResponseDelay = new Duration(transportconfig.RtpsWriter.NackResponseDelay.Val); //
       200 milliseconds
48             this.nackSuppressionDuration = new Duration(transportconfig.RtpsWriter.
       NackSuppressionDuration.Val);
49             this.pushMode = transportconfig.RtpsWriter.PushMode.Val;
50
51             InitTransmitters();
52             foreach (var trans in this.UDPTransmitters)
53             {
54                 trans.IsDiscovery = true;
55             }
56             this.Scheme = Encapsulation.PL_CDR_BE;
57
58             worker = new WriterWorker(this.PeriodicWork);
59         }
60
61         public void Start()
62         {
63             StartTransmitters();
64             worker.Start((int)this.ResendDataPeriod.AsMillis());
65         }
66
67         /// <summary>
68         /// The SPDPbuiltinParticipantWriter periodically sends this data-object to a pre-configured list
        of
69         /// locators to announce the Participant's presence on the network. This is achieved by
```

```
    periodically
70      /// calling StatelessWriter::unsent_changes_reset, which causes the StatelessWriter to resend all
71      /// changes present in its HistoryCache to all locators. The periodic rate at which the
72      /// SPDPbuiltinParticipantWriter sends out the SPDPdiscoveredParticipantData defaults to a PSM
    specified
73      /// Value. This period should be smaller than the leaseDuration specified in the
    SPDPdiscoveredParticipantData
74      /// </summary>
75      protected override void PeriodicWork()
76      {
77          foreach (var change in HistoryCache.Changes)
78          {
79              SendData(change);
80          }
81          this.UnsentChangesReset();
82      }
83
84      protected void SetLocatorListFromConfig(Transport transport, ParticipantImpl participant)
85      {
86          IList<Locator> unicastLocatorList = new List<Locator>();
87          IList<Locator> multicastLocatorList = new List<Locator>();
88
89          int PB = transport.Discovery.PortBase.Val;
90          int DG = transport.Discovery.DomainGain.Val;
91          int PG = transport.Discovery.ParticipantGain.Val;
92          int d0 = transport.Discovery.OffsetMetatrafficMulticast.Val;
93          int d1 = transport.Discovery.OffsetMetatrafficUnicast.Val;
94
95          string[] unicastStr = transport.Discovery.MetatrafficUnicastLocatorList.Val.Split(new char[]
    { ',', ';' }, StringSplitOptions.RemoveEmptyEntries);
96          foreach (string addr in unicastStr)
97          {
98              string[] parts = addr.Split(':');
99              IPAddress[] addresses = System.Net.Dns.GetHostAddresses(parts[0]);
100             IPAddress ipaddr = null;
101             if (addresses != null)
102             {
103                 foreach (var ipa in addresses)
104                     if (ipa.AddressFamily == System.Net.Sockets.AddressFamily.InterNetwork)
105                     {
106                         ipaddr = ipa;
107                         break;
108                     }
109             }
110             else
111                 throw new ArgumentException("Invalid unicast address " + parts[0]);
112
113             int port;
114             if (parts.Length <= 1)
115             {
116                 port = PB + DG * participant.DomainId + d1 + PG * participant.ParticipantId;
117             }
118             else
119                 port = int.Parse(parts[1]);
120
121             if (ipaddr != null)
122             {
123                 Locator locator = new Locator(ipaddr, port);
124                 log.DebugFormat("Using unicast Addr:{0} and Port:{0} for SPDP Discovery", ipaddr,
    port);
125                 unicastLocatorList.Add(locator);
126             }
127         }
128
129         string[] multicastStr = transport.Discovery.MetatrafficMulticastLocatorList.Val.Split(new
    char[] { ',', ';' }, StringSplitOptions.RemoveEmptyEntries);
130         foreach (string addr in multicastStr)
131         {
132             string[] parts = addr.Split(':');
133             IPAddress[] addresses = System.Net.Dns.GetHostAddresses(parts[0]);
134             IPAddress ipaddr = null;
135             if (addresses != null)
136             {
```

```
137                     foreach (var ipa in addresses)
138                         if (ipa.AddressFamily == System.Net.Sockets.AddressFamily.InterNetwork)
139                         {
140                             ipaddr = ipa;
141                             break;
142                         }
143                 }
144                 else
145                     throw new ArgumentException("Invalid multicast address " + parts[0]);
146 
147                 int port;
148                 if (parts.Length <= 1)
149                 {
150                     port = PB + DG * participant.DomainId + d0;
151                 }
152                 else
153                     port = int.Parse(parts[1]);
154                 if (ipaddr != null)
155                 {
156                     Locator locator = new Locator(ipaddr, port);
157                     log.DebugFormat("Using multicast Addr:{0} and Port:{0} for SPDP Discovery", ipaddr, ↙
    port);
158                     multicastLocatorList.Add(locator);
159                 }
160             }
161 
162             this.UnicastLocatorList = unicastLocatorList;
163             this.MulticastLocatorList = multicastLocatorList;
164         }
165     }
166 }
167 
```

```
1  using Doopec.Dds.Topic;
2
3  namespace Doopec.Rtps.Discovery
4  {
5      public class SPDPPublicationBuiltinTopicData : PublicationBuiltinTopicDataImpl
6      {
7          public const string PUBLICATION_TOPIC = "DCPSPublication,";
8
9          public SPDPPublicationBuiltinTopicData()
10         {
11             this.topicName = PUBLICATION_TOPIC;
12         }
13     }
14 }
15
```

```
1  using Doopec.Dds.Topic;
2
3  namespace Doopec.Rtps.Discovery
4  {
5      public class SPDPSubscriptionBuiltinTopicData : SubscriptionBuiltinTopicDataImpl
6      {
7          public const string SUBSCRIPTION_TOPIC = "DCPSSubscription";
8
9          public SPDPSubscriptionBuiltinTopicData()
10         {
11             this.topicName = SUBSCRIPTION_TOPIC;
12         }
13     }
14 }
15
```

```
1  using Doopec.Dds.Topic;
2
3  namespace Doopec.Rtps.Discovery
4  {
5      public class SPDPTopicBuiltinTopicData : TopicBuiltinTopicDataImpl
6      {
7          public const string TOPIC_TOPIC = "DCPSTopic,";
8
9          public SPDPTopicBuiltinTopicData()
10         {
11             this.topicName = TOPIC_TOPIC;
12         }
13     }
14 }
15
```

```
1  using Mina.Core.Buffer;
2  using Rtps.Messages.Submessages;
3  using System;
4  using System.Collections.Generic;
5  using System.Linq;
6  using System.Text;
7  using System.Threading.Tasks;
8
9  namespace Doopec.Rtps.Encoders
10 {
11     public static class AckNackEncoder
12     {
13         public static void PutAckNack(this IoBuffer buffer, AckNack obj)
14         {
15             buffer.PutEntityId(obj.ReaderId);
16             buffer.PutEntityId(obj.WriterId);
17             buffer.PutSequenceNumberSet(obj.ReaderSNState);
18             buffer.PutInt32(obj.Count);
19         }
20
21         public static AckNack GetAckNack(this IoBuffer buffer)
22         {
23             AckNack obj = new AckNack();
24             buffer.GetAckNack(ref obj);
25             return obj;
26         }
27
28         public static void GetAckNack(this IoBuffer buffer, ref AckNack obj)
29         {
30             obj.ReaderId = buffer.GetEntityId();
31             obj.WriterId = buffer.GetEntityId();
32             obj.ReaderSNState = buffer.GetSequenceNumberSet();
33             obj.Count = buffer.GetInt32();
34         }
35     }
36 }
37
```

```
using Mina.Core.Buffer;
using Rtps.Messages.Submessages;
using System;


namespace Doopec.Rtps.Encoders
{
    public static class DataFragEncoder
    {
        public static void PutDataFrag(this IoBuffer buffer, DataFrag obj)
        {
            buffer.PutInt16(obj.ExtraFlags);

            short octets_to_inline_qos = 4 + 4 + 8 + 4 + 2 + 2 + 4;
            buffer.PutInt16(octets_to_inline_qos);

            buffer.PutEntityId(obj.ReaderId);
            buffer.PutEntityId(obj.WriterId);
            buffer.PutSequenceNumber(obj.WriterSequenceNumber);

            buffer.PutInt32(obj.FragmentStartingNumber);
            buffer.PutInt16((short)obj.FragmentsInSubmessage);
            buffer.PutInt16((short)obj.FragmentSize);
            buffer.PutInt32(obj.SampleSize);

            if (obj.HasInlineQosFlag)
            {
                buffer.PutParameterList(obj.ParameterList);
            }

            buffer.Put(obj.SerializedPayload); // TODO: check this
        }

        public static DataFrag GetDataFrag(this IoBuffer buffer)
        {
            DataFrag obj = new DataFrag();
            buffer.GetDataFrag(ref obj);
            return obj;
        }

        public static void GetDataFrag(this IoBuffer buffer, ref DataFrag obj)
        {
            int start_count = buffer.Position; // start of bytes Read so far from the
            // beginning

            obj.ExtraFlags = (short)buffer.GetInt16();
            int octetsToInlineQos = buffer.GetInt16() & 0xffff;

            int currentCount = buffer.Position; // count bytes to inline qos

            obj.ReaderId = buffer.GetEntityId();
            obj.WriterId = buffer.GetEntityId();
            obj.WriterSequenceNumber = buffer.GetSequenceNumber();

            obj.FragmentStartingNumber = buffer.GetInt32(); // ulong
            obj.FragmentsInSubmessage = buffer.GetInt16(); // ushort
            obj.FragmentSize = buffer.GetInt16(); // ushort
            obj.SampleSize = buffer.GetInt32(); // ulong

            int bytesRead = buffer.Position - currentCount;
            int unknownOctets = octetsToInlineQos - bytesRead;

            for (int i = 0; i < unknownOctets; i++)
            {
                buffer.Get(); // Skip unknown octets, @see 9.4.5.3.3 octetsToInlineQos
            }

            if (obj.HasInlineQosFlag)
            {
                obj.ParameterList = buffer.GetParameterList();
            }

            int end_count = buffer.Position; // end of bytes Read so far from the beginning
            if (obj.Header.SubMessageLength != 0)
```

```
75              {
76                  obj.SerializedPayload = new byte[obj.Header.SubMessageLength - (end_count - start_count)];
77              }
78              else
79              { // SubMessage is the last one. Rest of the bytes are Read.
80                  // @see 8.3.3.2.3
81                  obj.SerializedPayload = new byte[buffer.Remaining];
82              }
83
84              buffer.Get(obj.SerializedPayload, 0, obj.SerializedPayload.Length);
85          }
86      }
87  }
88
```

```
1  using Mina.Core.Buffer;
2  using Rtps.Messages;
3  using Rtps.Messages.Submessages;
4  using Rtps.Messages.Submessages.Elements;
5  using System;
6  using Doopec.Utils.Network.Encoders;
7  using Doopec.Rtps.Messages;
8
9  namespace Doopec.Rtps.Encoders
10 {
11     public static class DataSubMessageEncoder
12     {
13         public static void PutDataSubMessage(this IoBuffer buffer, Data obj)
14         {
15             buffer.PutInt16(obj.ExtraFlags.Value);
16             short octetsToInlineQos = 0;
17             if (obj.HasInlineQosFlag)
18             {
19                 octetsToInlineQos = 4 + 4 + 8;// EntityId.LENGTH + EntityId.LENGTH + SequenceNumber.
    LENGTH;
20             }
21             buffer.PutInt16(octetsToInlineQos);
22             buffer.PutEntityId(obj.ReaderId);
23             buffer.PutEntityId(obj.WriterId);
24             buffer.PutSequenceNumber(obj.WriterSN);
25             if (obj.HasInlineQosFlag)
26             {
27                 buffer.PutParameterList(obj.InlineQos);
28             }
29
30             if (obj.HasDataFlag || obj.HasKeyFlag)
31             {
32                 buffer.Align(4);
33                 buffer.Put(obj.SerializedPayload.DataEncapsulation.SerializedPayload);
34             }
35         }
36
37         public static Data GetDataSubMessage(this IoBuffer buffer)
38         {
39             Data obj = new Data();
40             buffer.GetDataSubMessage(ref obj);
41             return obj;
42         }
43
44         public static void GetDataSubMessage(this IoBuffer buffer, ref Data obj)
45         {
46             if (obj.HasDataFlag && obj.HasKeyFlag)
47             {
48                 // Should we just ignore this message instead
49                 throw new ApplicationException(
50                         "This version of protocol does not allow Data submessage to contain both
    serialized data and serialized key (9.4.5.3.1)");
51             }
52
53             int start_count = buffer.Position; // start of bytes Read so far from the
54             // beginning
55             Flags flgs = new Flags();
56             flgs.Value = (byte)buffer.GetInt16();
57             obj.ExtraFlags = flgs;
58             int octetsToInlineQos = buffer.GetInt16() & 0xffff;
59
60             int currentCount = buffer.Position; // count bytes to inline qos
61
62             obj.ReaderId = buffer.GetEntityId();
63             obj.WriterId = buffer.GetEntityId();
64             obj.WriterSN = buffer.GetSequenceNumber();
65
66             int bytesRead = buffer.Position - currentCount;
67             int unknownOctets = octetsToInlineQos - bytesRead;
68
69             for (int i = 0; i < unknownOctets; i++)
70             {
71                 // TODO: Instead of looping, we should do just
72                 // newPos = bb.getBuffer.position() + unknownOctets or something
```

```
73                  // like that
74                  buffer.Get(); // Skip unknown octets, @see 9.4.5.3.3
75                  // octetsToInlineQos
76              }
77
78              if (obj.HasInlineQosFlag)
79              {
80                  obj.InlineQos = buffer.GetParameterList();
81              }
82
83              if (obj.HasDataFlag || obj.HasKeyFlag)
84              {
85                  buffer.Align(4); // Each submessage is aligned on 32-bit boundary, @see
86                  // 9.4.1 Overall Structure
87                  int end_count = buffer.Position; // end of bytes Read so far from the beginning
88                  int length;
89
90                  if (obj.Header.SubMessageLength != 0)
91                  {
92                      length =  obj.Header.SubMessageLength - (end_count - start_count);
93                  }
94                  else
95                  {
96                      // SubMessage is the last one. Rest of the bytes are Read.
97                      // @see 8.3.3.2.3
98                      length =  buffer.Remaining;
99                  }
100                 obj.SerializedPayload = new SerializedPayload();
101                 obj.SerializedPayload.DataEncapsulation = EncapsulationManager.Deserialize(buffer,
    length);
102             }
103
104
105         }
106     }
107 }
108
```

```
1  using Mina.Core.Buffer;
2  using Rtps.Messages;
3  using Rtps.Messages.Submessages.Elements;
4  using Rtps.Messages.Types;
5  using Rtps.Structure.Types;
6  using System;
7
8  namespace Doopec.Rtps.Encoders
9  {
10     public static class EncapsulationSchemeEncoder
11     {
12         public static void PutEncapsulationScheme(this IoBuffer buffer, EncapsulationScheme msg)
13         {
14             buffer.Put(msg.B0);
15             buffer.Put(msg.B1);
16             buffer.Put(msg.B2);
17             buffer.Put(msg.B3);
18         }
19
20         public static EncapsulationScheme GetEncapsulationScheme(this IoBuffer buffer)
21         {
22             EncapsulationScheme obj = new EncapsulationScheme();
23             obj.B0 = buffer.Get();
24             obj.B1 = buffer.Get();
25             obj.B2 = buffer.Get();
26             obj.B3 = buffer.Get();
27             return obj;
28         }
29
30         public static void GetEncapsulationScheme(this IoBuffer buffer, ref EncapsulationScheme obj)
31         {
32             obj.B0 = buffer.Get();
33             obj.B1 = buffer.Get();
34             obj.B2 = buffer.Get();
35             obj.B3 = buffer.Get();
36         }
37     }
38 }
39
```

```
1  using Doopec.Serializer;
2  using Mina.Core.Buffer;
3  using Rtps.Structure.Types;
4  using System.Reflection;
5
6  namespace Doopec.Rtps.Encoders
7  {
8      public static class EntityIdEncoder
9      {
10         public static void PutEntityId(this IoBuffer buffer, EntityId obj)
11         {
12             buffer.Put(obj.EntityKey);
13             buffer.Put((byte)obj.TypeID);
14         }
15         public static void WriteEntityId(IoBuffer buffer, EntityId obj)
16         {
17             buffer.Put(obj.EntityKey);
18             buffer.Put((byte)obj.TypeID);
19         }
20         public static EntityId GetEntityId(this IoBuffer buffer)
21         {
22             EntityId obj = new EntityId();
23             buffer.GetEntityId(ref obj);
24             return obj;
25         }
26
27         public static void GetEntityId(this IoBuffer buffer, ref EntityId obj)
28         {
29             buffer.Get(obj.EntityKey, 0, 3);
30             obj.TypeID = (EntityKinds)buffer.Get();
31         }
32         public static void ReadEntityId(IoBuffer buffer, ref EntityId obj)
33         {
34             if (obj == null)
35                 obj = new EntityId();
36
37             buffer.Get(obj.EntityKey, 0, 3);
38             obj.TypeID = (EntityKinds)buffer.Get();
39         }
40     }
41
42     public class EntityIdSerializer : IStaticTypeSerializer
43     {
44         delegate void WriterDelegate(IoBuffer buffer, EntityId obj);
45         delegate void ReaderDelegate(IoBuffer buffer, ref EntityId obj);
46
47         public void GetStaticMethods(System.Type type, out MethodInfo writer, out MethodInfo reader)
48         {
49             WriterDelegate writerDelegate = EntityIdEncoder.WriteEntityId;
50             ReaderDelegate readerDelegate = EntityIdEncoder.ReadEntityId;
51             writer = writerDelegate.Method;
52             reader = readerDelegate.Method;
53         }
54
55         public bool Handles(System.Type type)
56         {
57             return type == typeof(EntityId);
58         }
59
60         public System.Collections.Generic.IEnumerable<System.Type> GetSubtypes(System.Type type)
61         {
62             yield break;
63         }
64     }
65 }
66
```

```
1  using Mina.Core.Buffer;
2  using Rtps.Messages.Submessages;
3
4  namespace Doopec.Rtps.Encoders
5  {
6      public static class GapEncoder
7      {
8          public static void PutGap(this IoBuffer buffer, Gap obj)
9          {
10             buffer.PutEntityId(obj.ReaderId);
11             buffer.PutEntityId(obj.WriterId);
12             buffer.PutSequenceNumber(obj.GapStart);
13             buffer.PutSequenceNumberSet(obj.GapList);
14         }
15
16         public static Gap GetGap(this IoBuffer buffer)
17         {
18             Gap obj = new Gap();
19             buffer.GetGap(ref obj);
20             return obj;
21         }
22
23         public static void GetGap(this IoBuffer buffer, ref Gap obj)
24         {
25             obj.readerId = buffer.GetEntityId();
26             obj.writerId = buffer.GetEntityId();
27             obj.gapStart = buffer.GetSequenceNumber();
28             obj.gapList = buffer.GetSequenceNumberSet();
29         }
30     }
31 }
32
```

```
1  using Doopec.Serializer;
2  using Mina.Core.Buffer;
3  using Rtps.Structure.Types;
4  using System.Reflection;
5
6  namespace Doopec.Rtps.Encoders
7  {
8      public static class GUIDEncoder
9      {
10         public static void PutGUID(this IoBuffer buffer, GUID obj)
11         {
12             buffer.PutGuidPrefix(obj.Prefix);
13             buffer.PutEntityId(obj.EntityId);
14         }
15         public static void WriteGUID(IoBuffer buffer, GUID obj)
16         {
17             buffer.PutGuidPrefix(obj.Prefix);
18             buffer.PutEntityId(obj.EntityId);
19         }
20
21         public static GUID GetGUID(this IoBuffer buffer)
22         {
23             GUID obj = new GUID();
24             obj.Prefix = buffer.GetGuidPrefix();
25             obj.EntityId = buffer.GetEntityId();
26             return obj;
27         }
28
29         public static void GetGUID(this IoBuffer buffer, ref GUID obj)
30         {
31             obj.Prefix = buffer.GetGuidPrefix();
32             obj.EntityId = buffer.GetEntityId();
33         }
34
35         public static void ReadGUID(IoBuffer buffer, ref GUID obj)
36         {
37             if (obj == null)
38                 obj = new GUID();
39             obj.Prefix = buffer.GetGuidPrefix();
40             obj.EntityId = buffer.GetEntityId();
41         }
42     }
43
44     public class GUIDSerializer : IStaticTypeSerializer
45     {
46         delegate void WriterDelegate(IoBuffer buffer, GUID obj);
47         delegate void ReaderDelegate(IoBuffer buffer, ref GUID obj);
48
49         public void GetStaticMethods(System.Type type, out MethodInfo writer, out MethodInfo reader)
50         {
51             WriterDelegate writerDelegate = GUIDEncoder.WriteGUID;
52             ReaderDelegate readerDelegate = GUIDEncoder.ReadGUID;
53             writer = writerDelegate.Method;
54             reader = readerDelegate.Method;
55         }
56
57         public bool Handles(System.Type type)
58         {
59             return type == typeof(GUID);
60         }
61
62         public System.Collections.Generic.IEnumerable<System.Type> GetSubtypes(System.Type type)
63         {
64             yield break;
65         }
66     }
67 }
68
```

```
1  using Doopec.Serializer;
2  using Mina.Core.Buffer;
3  using Rtps.Structure.Types;
4  using System.Reflection;
5
6  namespace Doopec.Rtps.Encoders
7  {
8      public static class GuidPrefixEncoder
9      {
10         public static void PutGuidPrefix(this IoBuffer buffer, GuidPrefix obj)
11         {
12             buffer.Put(obj.Prefix);
13         }
14         public static void WriteGuidPrefix(IoBuffer buffer, GuidPrefix obj)
15         {
16             buffer.Put(obj.Prefix);
17         }
18
19         public static GuidPrefix GetGuidPrefix(this IoBuffer buffer)
20         {
21             GuidPrefix obj = new GuidPrefix();
22             buffer.GetGuidPrefix(ref obj);
23             return obj;
24         }
25
26         public static void GetGuidPrefix(this IoBuffer buffer, ref GuidPrefix obj)
27         {
28             buffer.Get(obj.Prefix, 0, GuidPrefix.GUID_PREFIX_SIZE);
29         }
30
31         public static void ReadGuidPrefix(IoBuffer buffer, ref GuidPrefix obj)
32         {
33             buffer.Get(obj.Prefix, 0, GuidPrefix.GUID_PREFIX_SIZE);
34         }
35     }
36
37     public class GuidPrefixSerializer : IStaticTypeSerializer
38     {
39         delegate void WriterDelegate(IoBuffer buffer, GuidPrefix obj);
40         delegate void ReaderDelegate(IoBuffer buffer, ref GuidPrefix obj);
41
42         public void GetStaticMethods(System.Type type, out MethodInfo writer, out MethodInfo reader)
43         {
44             WriterDelegate writerDelegate = GuidPrefixEncoder.WriteGuidPrefix;
45             ReaderDelegate readerDelegate = GuidPrefixEncoder.ReadGuidPrefix;
46             writer = writerDelegate.Method;
47             reader = readerDelegate.Method;
48         }
49
50         public bool Handles(System.Type type)
51         {
52             return type == typeof(GuidPrefix);
53         }
54
55         public System.Collections.Generic.IEnumerable<System.Type> GetSubtypes(System.Type type)
56         {
57             yield break;
58         }
59     }
60 }
61
```

```
using Mina.Core.Buffer;
using Rtps.Messages;
using Rtps.Messages.Types;
using Rtps.Structure.Types;
using System;

namespace Doopec.Rtps.Encoders
{
    public static class HeaderEncoder
    {
        public static void PutHeader(this IoBuffer buffer, Header msg)
        {
            buffer.PutProtocolId(msg.Protocol);
            buffer.PutProtocolVersion(msg.Version);
            buffer.PutVendorId(msg.VendorId);
            buffer.PutGuidPrefix(msg.GuidPrefix);
        }

        public static Header GetHeader(this IoBuffer buffer)
        {
            Header obj = new Header();
            buffer.GetHeader(ref obj);
            return obj;
        }

        public static void GetHeader(this IoBuffer buffer, ref Header obj)
        {
            ProtocolId protocol = buffer.GetProtocolId();
            if (!protocol.Equals(ProtocolId.PROTOCOL_RTPS))
            {
                throw new ApplicationException("Wrong Protocol ID. Expected RTPS");
            }
            ProtocolVersion version = buffer.GetProtocolVersion();
            if (!version.Equals(ProtocolVersion.PROTOCOLVERSION))
            {
                throw new ApplicationException("Wrong Protocol version. Expected " + ProtocolVersion.
    PROTOCOLVERSION);
            }
            obj.VendorId = buffer.GetVendorId();
            obj.GuidPrefix = buffer.GetGuidPrefix();
        }
    }
}

```

```
1  using Mina.Core.Buffer;
2  using Rtps.Messages.Submessages;
3
4  namespace Doopec.Rtps.Encoders
5  {
6      public static class HeartbeatEncoder
7      {
8          public static void PutHeartbeat(this IoBuffer buffer, Heartbeat obj)
9          {
10             buffer.PutEntityId(obj.readerId);
11             buffer.PutEntityId(obj.writerId);
12             buffer.PutSequenceNumber(obj.firstSN);
13             buffer.PutSequenceNumber(obj.lastSN);
14             buffer.PutInt32(obj.count);
15         }
16
17         public static Heartbeat GetHeartbeat(this IoBuffer buffer)
18         {
19             Heartbeat obj = new Heartbeat();
20             buffer.GetHeartbeat(ref obj);
21             return obj;
22         }
23
24         public static void GetHeartbeat(this IoBuffer buffer, ref Heartbeat obj)
25         {
26             obj.readerId = buffer.GetEntityId();
27             obj.writerId = buffer.GetEntityId();
28             obj.firstSN = buffer.GetSequenceNumber();
29             obj.lastSN = buffer.GetSequenceNumber();
30             obj.count = buffer.GetInt32();
31         }
32     }
33 }
34
```

```
1  using Mina.Core.Buffer;
2  using Rtps.Messages.Submessages;
3
4  namespace Doopec.Rtps.Encoders
5  {
6      public static class HeartbeatFragEncoder
7      {
8          public static void PutHeartbeatFrag(this IoBuffer buffer, HeartbeatFrag obj)
9          {
10             buffer.PutEntityId(obj.ReaderId);
11             buffer.PutEntityId(obj.WriterId);
12             buffer.PutSequenceNumber(obj.WriterSequenceNumber);
13             buffer.PutInt32(obj.LastFragmentNumber);
14             buffer.PutInt32(obj.Count);
15         }
16
17         public static HeartbeatFrag GetHeartbeatFrag(this IoBuffer buffer)
18         {
19             HeartbeatFrag obj = new HeartbeatFrag();
20             buffer.GetHeartbeatFrag(ref obj);
21             return obj;
22         }
23
24         public static void GetHeartbeatFrag(this IoBuffer buffer, ref HeartbeatFrag obj)
25         {
26             obj.ReaderId = buffer.GetEntityId();
27             obj.WriterId = buffer.GetEntityId();
28             obj.WriterSequenceNumber = buffer.GetSequenceNumber();
29             obj.LastFragmentNumber = buffer.GetInt32();
30             obj.Count = buffer.GetInt32();
31         }
32     }
33 }
34
```

```
1  using Mina.Core.Buffer;
2  using Rtps.Messages.Submessages;
3
4  namespace Doopec.Rtps.Encoders
5  {
6      public static class InfoDestinationEncoder
7      {
8          public static void PutInfoDestination(this IoBuffer buffer, InfoDestination msg)
9          {
10             buffer.PutGuidPrefix(msg.GuidPrefix);
11         }
12
13         public static InfoDestination GetInfoDestination(this IoBuffer buffer)
14         {
15             InfoDestination obj = new InfoDestination();
16             buffer.GetInfoDestination(ref obj);
17             return obj;
18         }
19
20         public static void GetInfoDestination(this IoBuffer buffer, ref InfoDestination obj)
21         {
22             obj.GuidPrefix = buffer.GetGuidPrefix();
23         }
24     }
25 }
26
```

```
1  using Mina.Core.Buffer;
2  using Rtps.Messages.Submessages;
3  using Rtps.Structure.Types;
4
5  namespace Doopec.Rtps.Encoders
6  {
7      public static class InfoReplyEncoder
8      {
9          public static void PutInfoReply(this IoBuffer buffer, InfoReply obj)
10         {
11             buffer.PutInt64(obj.UnicastLocatorList.Count);
12             foreach (Locator loc in obj.UnicastLocatorList)
13             {
14                 buffer.PutLocator(loc);
15             }
16
17             if (obj.HasMulticastFlag)
18             {
19                 buffer.PutInt64(obj.MulticastLocatorList.Count);
20                 foreach (Locator loc in obj.MulticastLocatorList)
21                 {
22                     buffer.PutLocator(loc);
23                 }
24             }
25         }
26
27         public static InfoReply GetInfoReply(this IoBuffer buffer)
28         {
29             InfoReply obj = new InfoReply();
30             buffer.GetInfoReply(ref obj);
31             return obj;
32         }
33
34         public static void GetInfoReply(this IoBuffer buffer, ref InfoReply obj)
35         {
36             long numLocators = buffer.GetInt64();
37             //log.DebugFormat("Reading {}(0x{}) locators", numLocators, string.Format("%08x",
       numLocators));
38             for (int i = 0; i < numLocators; i++)
39             {
40                 Locator loc = buffer.GetLocator();
41
42                 obj.UnicastLocatorList.Add(loc);
43             }
44
45             if (obj.HasMulticastFlag)
46             {
47                 numLocators = buffer.GetInt64();
48                 for (int i = 0; i < numLocators; i++)
49                 {
50                     Locator loc = buffer.GetLocator();
51
52                     obj.MulticastLocatorList.Add(loc);
53                 }
54             }
55         }
56     }
57 }
58
```

```
1  using Mina.Core.Buffer;
2  using Rtps.Messages.Submessages;
3  using System;
4  using System.Collections.Generic;
5  using System.Linq;
6  using System.Text;
7  using System.Threading.Tasks;
8
9  namespace Doopec.Rtps.Encoders
10 {
11     public static class InfoReplyIp4Encoder
12     {
13         public static void PutInfoReplyIp4(this IoBuffer buffer, InfoReplyIp4 obj)
14         {
15             buffer.PutLocatorUDPv4(obj.UnicastLocator);
16             buffer.PutLocatorUDPv4(obj.MulticastLocator);
17         }
18
19         public static InfoReplyIp4 GetInfoReplyIp4(this IoBuffer buffer)
20         {
21             InfoReplyIp4 obj = new InfoReplyIp4();
22             buffer.GetInfoReplyIp4(ref obj);
23             return obj;
24         }
25
26         public static void GetInfoReplyIp4(this IoBuffer buffer, ref InfoReplyIp4 obj)
27         {
28             obj.UnicastLocator = buffer.GetLocatorUDPv4();
29             obj.MulticastLocator = buffer.GetLocatorUDPv4();
30         }
31     }
32 }
33
```

```
1  using Mina.Core.Buffer;
2  using Rtps.Messages.Submessages;
3
4  namespace Doopec.Rtps.Encoders
5  {
6      public static class InfoSourceEncoder
7      {
8          public static void PutInfoSource(this IoBuffer buffer, InfoSource obj)
9          {
10             buffer.PutInt64(0);
11             buffer.PutProtocolVersion(obj.ProtocolVersion);
12             buffer.PutVendorId(obj.VendorId);
13             buffer.PutGuidPrefix(obj.GuidPrefix);
14         }
15
16         public static InfoSource GetInfoSource(this IoBuffer buffer)
17         {
18             InfoSource obj = new InfoSource();
19             buffer.GetInfoSource(ref obj);
20             return obj;
21         }
22
23         public static void GetInfoSource(this IoBuffer buffer, ref InfoSource obj)
24         {
25             buffer.GetInt64(); // unused
26
27             obj.ProtocolVersion = buffer.GetProtocolVersion();
28             obj.VendorId = buffer.GetVendorId();
29             obj.GuidPrefix = buffer.GetGuidPrefix();
30         }
31     }
32 }
33
```

```
1  using Mina.Core.Buffer;
2  using Rtps.Messages.Submessages;
3
4  namespace Doopec.Rtps.Encoders
5  {
6      public static class InfoTimestampEncoder
7      {
8          public static void PutInfoTimestamp(this IoBuffer buffer, InfoTimestamp obj)
9          {
10             if (!obj.HasInvalidateFlag)
11             {
12                 buffer.PutTime(obj.TimeStamp);
13             }
14         }
15
16         public static InfoTimestamp GetInfoTimestamp(this IoBuffer buffer)
17         {
18             InfoTimestamp obj = new InfoTimestamp();
19             buffer.GetInfoTimestamp(ref obj);
20             return obj;
21         }
22
23         public static void GetInfoTimestamp(this IoBuffer buffer, ref InfoTimestamp obj)
24         {
25             if (!obj.HasInvalidateFlag)
26             {
27                 obj.TimeStamp = buffer.GetTime();
28             }
29
30         }
31     }
32 }
33
```

```
using Doopec.Serializer;
using Mina.Core.Buffer;
using Rtps.Structure.Types;
using System;
using System.Reflection;

namespace Doopec.Rtps.Encoders
{
    public static class LocatorEncoder
    {
        public static void PutLocator(this IoBuffer buffer, Locator obj)
        {
            buffer.PutInt32((int)obj.Kind);
            buffer.PutInt32(obj.Port);
            buffer.Put(obj.SocketAddressBytes);
        }

        public static void WriteLocator(IoBuffer buffer, Locator obj)
        {
            buffer.PutInt32((int)obj.Kind);
            buffer.PutInt32(obj.Port);
            buffer.Put(obj.SocketAddressBytes);
        }

        public static Locator GetLocator(this IoBuffer buffer)
        {
            Locator obj = new Locator();
            buffer.GetLocator(ref obj);
            return obj;
        }

        public static void GetLocator(this IoBuffer buffer, ref Locator obj)
        {
            obj.Kind = (LocatorKind)buffer.GetInt32();
            obj.Port = (int)buffer.GetInt32(); ;
            byte[] tmp = new byte[16];

            buffer.Get(tmp, 0, 16);
            obj.SocketAddressBytes = tmp;
        }

        public static void ReadLocator(IoBuffer buffer, ref Locator obj)
        {
            if (obj == null)
                obj = new Locator();
            obj.Kind = (LocatorKind)buffer.GetInt32();
            obj.Port = (int)buffer.GetInt32(); ;
            byte[] tmp = new byte[16];

            buffer.Get(tmp, 0, 16);
            obj.SocketAddressBytes = tmp;
        }
    }
    public class LocatorSerializer : IStaticTypeSerializer
    {
        delegate void WriterDelegate(IoBuffer buffer, Locator obj);
        delegate void ReaderDelegate(IoBuffer buffer, ref Locator obj);

        public void GetStaticMethods(System.Type type, out MethodInfo writer, out MethodInfo reader)
        {
            WriterDelegate writerDelegate = LocatorEncoder.WriteLocator;
            ReaderDelegate readerDelegate = LocatorEncoder.ReadLocator;
            writer = writerDelegate.Method;
            reader = readerDelegate.Method;
        }

        public bool Handles(System.Type type)
        {
            return type == typeof(Locator);
        }

        public System.Collections.Generic.IEnumerable<System.Type> GetSubtypes(System.Type type)
        {
            yield break;
```

```
75          }
76      }
77  }
78
```

```
1  using Mina.Core.Buffer;
2  using Rtps.Messages.Submessages.Elements;
3
4  namespace Doopec.Rtps.Encoders
5  {
6      public static class LocatorUDPv4Encoder
7      {
8          public static void PutLocatorUDPv4(this IoBuffer buffer, LocatorUDPv4 obj)
9          {
10             buffer.PutInt32(obj.Port);
11             buffer.PutInt32(obj.Address);
12         }
13
14         public static LocatorUDPv4 GetLocatorUDPv4(this IoBuffer buffer)
15         {
16             LocatorUDPv4 obj = new LocatorUDPv4();
17             buffer.GetLocatorUDPv4(ref obj);
18             return obj;
19         }
20
21         public static void GetLocatorUDPv4(this IoBuffer buffer, ref LocatorUDPv4 obj)
22         {
23             obj.Port = buffer.GetInt32();
24             obj.Address = buffer.GetInt32();
25         }
26     }
27 }
28
```

```
using Mina.Core.Buffer;
using Mina.Core.Session;
using Mina.Filter.Codec;
using Rtps.Messages;
using System;

namespace Doopec.Rtps.Encoders
{
    public class MessageEncoder : IProtocolEncoder
    {

        public void Encode(IoSession session, object message, IProtocolEncoderOutput output)
        {
            Message msg = (Message)message;
            IoBuffer buffer = IoBuffer.Allocate(1024);
            buffer.AutoExpand = true;
            buffer.PutMessage(msg);
            buffer.Flip();
            output.Write(buffer);
        }

        public void Dispose(IoSession session)
        {
            // nothing to Dispose
        }
    }

    public class MessageDecoder : CumulativeProtocolDecoder
    {
        protected override Boolean DoDecode(IoSession session, IoBuffer input, IProtocolDecoderOutput ↙
    output)
        {
            if (input.Remaining >= 4)
            {
                Message request = input.GetMessage();
                output.Write(request);
                return true;
            }
            else
            {
                return false;
            }
        }
    }

    public class MessageCodecFactory : IProtocolCodecFactory
    {
        public static MessageEncoder encoder = new MessageEncoder();
        public static MessageDecoder decoder = new MessageDecoder();

        public IProtocolEncoder GetEncoder(IoSession session)
        {
            return encoder;
        }

        public IProtocolDecoder GetDecoder(IoSession session)
        {
            return decoder;
        }
    }
}

```

```
1  using Mina.Core.Buffer;
2  using Rtps.Messages;
3  using System;
4  using System.Collections.Generic;
5  using System.Linq;
6  using System.Text;
7  using System.Threading.Tasks;
8
9  namespace Doopec.Rtps.Encoders
10 {
11     public static class MessageStaticEncoder
12     {
13         public static void PutMessage(this IoBuffer buffer, Message msg)
14         {
15             buffer.PutHeader(msg.Header);
16             int position = 0;
17             int subMessageCount = 0;
18             foreach (SubMessage submsg in msg.SubMessages)
19             {
20                 int subMsgStartPosition = buffer.Position;
21                 position = buffer.PutSubMessage(submsg);
22                 subMessageCount++;
23             }
24             // Length of last submessage is 0, @see 8.3.3.2.3 submessageLength
25             if (subMessageCount > 0)
26                 buffer.PutInt16(position - 2, (short)0);
27         }
28
29         public static Message GetMessage(this IoBuffer buffer)
30         {
31             Message obj = new Message();
32             buffer.GetMessage(ref obj);
33             return obj;
34         }
35
36         public static void GetMessage(this IoBuffer buffer, ref Message obj)
37         {
38             ByteOrder byteOrder = buffer.Order;
39             buffer.Order = ByteOrder.LittleEndian;
40             obj.Header = buffer.GetHeader();
41             while (buffer.HasRemaining)
42             {
43                 SubMessage submsg = buffer.GetSubMessage();
44                 obj.SubMessages.Add(submsg);
45             }
46             buffer.Order = byteOrder;
47         }
48     }
49 }
50
```

```
1  using Mina.Core.Buffer;
2  using Rtps.Messages.Submessages;
3
4  namespace Doopec.Rtps.Encoders
5  {
6      public static class NackFragEncoder
7      {
8          public static void PutNackFrag(this IoBuffer buffer, NackFrag obj)
9          {
10             buffer.PutEntityId(obj.ReaderId);
11             buffer.PutEntityId(obj.WriterId);
12             buffer.PutSequenceNumber(obj.WriterSequenceNumber);
13             buffer.PutSequenceNumberSet(obj.FragmentNumberState);
14             buffer.PutInt32(obj.Count);
15         }
16
17         public static NackFrag GetNackFrag(this IoBuffer buffer)
18         {
19             NackFrag obj = new NackFrag();
20             buffer.GetNackFrag(ref obj);
21             return obj;
22         }
23
24         public static void GetNackFrag(this IoBuffer buffer, ref NackFrag obj)
25         {
26             obj.ReaderId = buffer.GetEntityId();
27             obj.WriterId = buffer.GetEntityId();
28             obj.WriterSequenceNumber = buffer.GetSequenceNumber();
29             obj.FragmentNumberState = buffer.GetSequenceNumberSet();
30             obj.Count = buffer.GetInt32();
31         }
32     }
33 }
34
```

```
1  using Mina.Core.Buffer;
2  using Rtps.Messages.Submessages;
3
4  namespace Doopec.Rtps.Encoders
5  {
6      public static class PadEncoder
7      {
8          public static void PutPad(this IoBuffer buffer, Pad obj)
9          {
10             buffer.Put(obj.Bytes);
11         }
12
13         public static Pad GetPad(this IoBuffer buffer)
14         {
15             Pad obj = new Pad();
16             buffer.GetPad(ref obj);
17             return obj;
18         }
19
20         public static void GetPad(this IoBuffer buffer, ref Pad obj)
21         {
22             obj.Bytes = new byte[buffer.Remaining];
23             buffer.Get(obj.Bytes, 0, obj.Bytes.Length);
24         }
25     }
26 }
27
```

```
1  using Mina.Core.Buffer;
2  using Rtps.Messages.Submessages.Elements;
3  using Rtps.Messages.Types;
4  using Doopec.Utils.Network.Encoders;
5
6  namespace Doopec.Rtps.Encoders
7  {
8      public static class ParameterEncoder
9      {
10         public static void PutParameter(this IoBuffer buffer, Parameter obj)
11         {
12             buffer.PutInt16((short)obj.ParameterId);
13             buffer.PutInt16(0); // length will be calculated
14
15             int pos = buffer.Position;
16             buffer.Put(obj.Bytes);
17
18             buffer.Align(4); // Make sure length is multiple of 4 & align for
19             // next param
20
21             int paramLength = buffer.Position - pos;
22             buffer.PutInt16(pos - 2, (short)paramLength);
23         }
24
25         public static Parameter GetParameter(this IoBuffer buffer)
26         {
27             Parameter obj = new Parameter();
28             buffer.GetParameter(ref obj);
29             return obj;
30         }
31
32         public static void GetParameter(this IoBuffer buffer, ref Parameter obj)
33         {
34             obj.ParameterId = (ParameterId)buffer.GetInt16();
35             int length = buffer.GetInt16();
36             obj.Bytes = new byte[length];
37             buffer.Get(obj.Bytes, 0, length);
38         }
39     }
40 }
41
```

```
1  using log4net;
2  using Mina.Core.Buffer;
3  using System;
4  using System.Collections.Generic;
5  using System.Linq;
6  using System.Reflection;
7  using System.Text;
8  using System.Threading.Tasks;
9  using Rtps.Messages.Submessages.Elements;
10 using Rtps.Messages.Types;
11 using Doopec.Encoders.RTPS;
12 using Doopec.Utils.Network.Encoders;
13
14 namespace Doopec.Rtps.Encoders
15 {
16     public static class ParameterListEncoder
17     {
18         private static readonly ILog log = LogManager.GetLogger(MethodBase.GetCurrentMethod().
       DeclaringType);
19
20         public static void PutParameterList(this IoBuffer buffer, ParameterList obj)
21         {
22             buffer.Align(4); // @see 9.4.2.11
23
24             obj.Value.Add(Sentinel.Instance); // Sentinel must be the last Parameter
25             foreach (Parameter param in obj.Value)
26             {
27                 buffer.PutParameter(param);
28             }
29         }
30
31         public static ParameterList GetParameterList(this IoBuffer buffer)
32         {
33             ParameterList obj = new ParameterList();
34             buffer.GetParameterList(ref obj);
35             return obj;
36         }
37
38         public static void GetParameterList(this IoBuffer buffer, ref ParameterList obj)
39         {
40             log.Debug("Reading ParameterList from buffer");
41
42             while (true)
43             {
44                 int pos1 = buffer.Position;
45
46                 Parameter param = buffer.GetParameter();
47                 obj.Value.Add(param);
48                 int length = buffer.Position - pos1;
49
50                 //log.DebugFormat("Read Parameter {0}, length {1} from position {2}", param, length, pos1)
       ;
51
52                 if (param.ParameterId == ParameterId.PID_SENTINEL)
53                 {
54                     break;
55                 }
56             }
57         }
58     }
59 }
60
```

```
using Doopec.Serializer;
using Mina.Core.Buffer;
using Rtps.Messages.Types;
using System.Reflection;

namespace Doopec.Rtps.Encoders
{
    public static class ProtocolIdEncoder
    {
        public static void PutProtocolId(this IoBuffer buffer, ProtocolId obj)
        {
            buffer.Put(obj.Id);
        }
        public static void WriteProtocolId(IoBuffer buffer, ProtocolId obj)
        {
            buffer.Put(obj.Id);
        }
        public static ProtocolId GetProtocolId(this IoBuffer buffer)
        {
            ProtocolId obj = new ProtocolId();
            buffer.GetProtocolId(ref obj);
            return obj;
        }

        public static void GetProtocolId(this IoBuffer buffer, ref ProtocolId obj)
        {
            buffer.Get(obj.Id, 0, ProtocolId.PROTOID_SIZE);
        }
        public static void ReadProtocolId( IoBuffer buffer, ref ProtocolId obj)
        {
            if (obj == null)
                obj = new ProtocolId();
            buffer.Get(obj.Id, 0, ProtocolId.PROTOID_SIZE);
        }
    }
    public class ProtocolIdSerializer : IStaticTypeSerializer
    {
        delegate void WriterDelegate(IoBuffer buffer, ProtocolId obj);
        delegate void ReaderDelegate(IoBuffer buffer, ref ProtocolId obj);

        public void GetStaticMethods(System.Type type, out MethodInfo writer, out MethodInfo reader)
        {
            WriterDelegate writerDelegate = ProtocolIdEncoder.WriteProtocolId;
            ReaderDelegate readerDelegate = ProtocolIdEncoder.ReadProtocolId;
            writer = writerDelegate.Method;
            reader = readerDelegate.Method;
        }

        public bool Handles(System.Type type)
        {
            return type == typeof(ProtocolId);
        }

        public System.Collections.Generic.IEnumerable<System.Type> GetSubtypes(System.Type type)
        {
            yield break;
        }
    }
}

```

```
1  using Doopec.Serializer;
2  using Mina.Core.Buffer;
3  using Rtps.Structure.Types;
4  using System.Reflection;
5
6  namespace Doopec.Rtps.Encoders
7  {
8      public static class ProtocolVersionEncoder
9      {
10         public static void PutProtocolVersion(this IoBuffer buffer, ProtocolVersion obj)
11         {
12             buffer.Put(obj.Major);
13             buffer.Put(obj.Minor);
14         }
15         public static void WriteProtocolVersion(IoBuffer buffer, ProtocolVersion obj)
16         {
17             buffer.Put(obj.Major);
18             buffer.Put(obj.Minor);
19         }
20
21         public static ProtocolVersion GetProtocolVersion(this IoBuffer buffer)
22         {
23             ProtocolVersion obj = new ProtocolVersion();
24             buffer.GetProtocolVersion(ref obj);
25             return obj;
26         }
27
28         public static void GetProtocolVersion(this IoBuffer buffer, ref ProtocolVersion obj)
29         {
30             obj.Major = buffer.Get();
31             obj.Minor = buffer.Get();
32         }
33         public static void ReadProtocolVersion( IoBuffer buffer, ref ProtocolVersion obj)
34         {
35             if (obj == null)
36                 obj = new ProtocolVersion();
37             obj.Major = buffer.Get();
38             obj.Minor = buffer.Get();
39         }
40     }
41     public class ProtocolVersionSerializer : IStaticTypeSerializer
42     {
43         delegate void WriterDelegate(IoBuffer buffer, ProtocolVersion obj);
44         delegate void ReaderDelegate(IoBuffer buffer, ref ProtocolVersion obj);
45
46         public void GetStaticMethods(System.Type type, out MethodInfo writer, out MethodInfo reader)
47         {
48             WriterDelegate writerDelegate = ProtocolVersionEncoder.WriteProtocolVersion;
49             ReaderDelegate readerDelegate = ProtocolVersionEncoder.ReadProtocolVersion;
50             writer = writerDelegate.Method;
51             reader = readerDelegate.Method;
52         }
53
54         public bool Handles(System.Type type)
55         {
56             return type == typeof(ProtocolVersion);
57         }
58
59         public System.Collections.Generic.IEnumerable<System.Type> GetSubtypes(System.Type type)
60         {
61             yield break;
62         }
63     }
64 }
65
```

```
1  using Rtps.Messages.Submessages.Elements;
2  using Rtps.Messages.Types;
3
4  namespace Doopec.Encoders.RTPS
5  {
6      public class Sentinel : Parameter
7      {
8          private static Sentinel instance = new Sentinel();
9          private Sentinel()
10             : base(ParameterId.PID_SENTINEL)
11         {
12             this.Bytes = new byte[0];
13         }
14
15         public static Sentinel Instance { get { return instance; } }
16     }
17 }
18
```

```
1  using Mina.Core.Buffer;
2  using Rtps.Structure.Types;
3
4  namespace Doopec.Rtps.Encoders
5  {
6      public static class SequenceNumberEncoder
7      {
8          public static void PutSequenceNumber(this IoBuffer buffer, SequenceNumber obj)
9          {
10             buffer.PutInt32(obj.High);
11             buffer.PutInt32((int)obj.Low);
12         }
13
14         public static SequenceNumber GetSequenceNumber(this IoBuffer buffer)
15         {
16             SequenceNumber obj = new SequenceNumber();
17             buffer.GetSequenceNumber(ref obj);
18             return obj;
19         }
20
21         public static void GetSequenceNumber(this IoBuffer buffer, ref SequenceNumber obj)
22         {
23             obj.High = buffer.GetInt32();
24             obj.Low = (uint)buffer.GetInt32(); ;
25         }
26     }
27 }
28
```

```
1  using Mina.Core.Buffer;
2   using Rtps.Messages.Submessages.Elements;
3  using System;
4  using System.Collections.Generic;
5  using System.Linq;
6  using System.Text;
7  using System.Threading.Tasks;
8
9  namespace Doopec.Rtps.Encoders
10 {
11     public static class SequenceNumberSetEncoder
12     {
13         public static void PutSequenceNumberSet(this IoBuffer buffer, SequenceNumberSet obj)
14         {
15             buffer.PutSequenceNumber(obj.BitmapBase);
16
17             // buffer.write_long(bitmaps.length);
18             // buffer.write_long(bitmaps.length * 32);
19             buffer.PutInt32(obj.NumBits);
20             for (int i = 0; i < obj.Bitmaps.Length; i++)
21             {
22                 buffer.PutInt32(obj.Bitmaps[i]);
23             }
24         }
25
26         public static SequenceNumberSet GetSequenceNumberSet(this IoBuffer buffer)
27         {
28             SequenceNumberSet obj = new SequenceNumberSet();
29             buffer.GetSequenceNumberSet(ref obj);
30             return obj;
31         }
32
33         public static void GetSequenceNumberSet(this IoBuffer buffer, ref SequenceNumberSet obj)
34         {
35             obj.BitmapBase = buffer.GetSequenceNumber();
36
37             obj.NumBits = buffer.GetInt32();
38             int count = (obj.NumBits + 31) / 32;
39             obj.Bitmaps = new int[count];
40
41             for (int i = 0; i < obj.Bitmaps.Length; i++)
42             {
43                 obj.Bitmaps[i] = buffer.GetInt32();
44             }
45         }
46     }
47 }
48
```

```
1  using Mina.Core.Buffer;
2  using rtps.messages.elements;
3  using System;
4  using System.Collections.Generic;
5  using System.Linq;
6  using System.Text;
7  using System.Threading.Tasks;
8
9  namespace Doopec.Rtps.Encoders
10 {
11     public static class StatusInfoEncoder
12     {
13         public static void PutStatusInfo(this IoBuffer buffer, StatusInfo obj)
14         {
15             buffer.Put(obj.Bytes);
16         }
17
18         public static StatusInfo GetStatusInfo(this IoBuffer buffer)
19         {
20             StatusInfo obj = new StatusInfo();
21             buffer.GetStatusInfo(ref obj);
22             return obj;
23         }
24
25         public static void GetStatusInfo(this IoBuffer buffer, ref StatusInfo obj)
26         {
27              buffer.Get(obj.Bytes, 0, 4);
28         }
29     }
30 }
31
```

```
1  using Mina.Core.Buffer;
2  using System;
3  using System.Collections.Generic;
4  using System.Linq;
5  using System.Text;
6  using System.Threading.Tasks;
7  using System.Reflection;
8  using log4net;
9  using Rtps.Messages;
10 using Rtps.Messages.Types;
11 using Rtps.Messages.Submessages;
12 using Doopec.Utils.Network.Encoders;
13
14 namespace Doopec.Rtps.Encoders
15 {
16     public static class SubMessageEncoder
17     {
18         private static readonly ILog log = LogManager.GetLogger(MethodBase.GetCurrentMethod().
       DeclaringType);
19
20         public static int PutSubMessage(this IoBuffer buffer, SubMessage msg)
21         {
22             // The PSM aligns each Submessage on a 32-bit boundary with respect to the start of the
       Message (page 159).
23             buffer.Align(4);
24             buffer.Order = (msg.Header.IsLittleEndian ? ByteOrder.LittleEndian : ByteOrder.BigEndian); //
        Set the endianess
25             buffer.PutSubMessageHeader(msg.Header);
26             int position = buffer.Position;
27             switch (msg.Kind)
28             {
29                 case SubMessageKind.PAD:
30                     buffer.PutPad((Pad)msg);
31                     break;
32                 case SubMessageKind.ACKNACK:
33                     buffer.PutAckNack((AckNack)msg);
34                     break;
35                 case SubMessageKind.HEARTBEAT:
36                     buffer.PutHeartbeat((Heartbeat)msg);
37                     break;
38                 case SubMessageKind.GAP:
39                     buffer.PutGap((Gap)msg);
40                     break;
41                 case SubMessageKind.INFO_TS:
42                     buffer.PutInfoTimestamp((InfoTimestamp)msg);
43                     break;
44                 case SubMessageKind.INFO_SRC:
45                     buffer.PutInfoSource((InfoSource)msg);
46                     break;
47                 case SubMessageKind.INFO_REPLY_IP4:
48                     buffer.PutInfoReplyIp4((InfoReplyIp4)msg);
49                     break;
50                 case SubMessageKind.INFO_DST:
51                     buffer.PutInfoDestination((InfoDestination)msg);
52                     break;
53                 case SubMessageKind.INFO_REPLY:
54                     buffer.PutInfoReply((InfoReply)msg);
55                     break;
56                 case SubMessageKind.NACK_FRAG:
57                     buffer.PutNackFrag((NackFrag)msg);
58                     break;
59                 case SubMessageKind.HEARTBEAT_FRAG:
60                     buffer.PutHeartbeatFrag((HeartbeatFrag)msg);
61                     break;
62                 case SubMessageKind.DATA:
63                     buffer.PutDataSubMessage((Data)msg);
64                     break;
65                 case SubMessageKind.DATA_FRAG:
66                     buffer.PutDataFrag((DataFrag)msg);
67                     break;
68                 default:
69                     break;
70             }
71             buffer.Align(4);
```

```
72              int subMessageLength = buffer.Position - position;
73
74              // Position to 'submessageLength' -2 is for short (2 bytes)
75              // buffers current position is not changed
76              buffer.PutInt16(position - 2, (short)subMessageLength);
77              return position;
78          }
79
80          public static SubMessage GetSubMessage(this IoBuffer buffer)
81          {
82              SubMessage obj = new SubMessage();
83              buffer.GetSubMessage(ref obj);
84              return obj;
85          }
86
87          public static void GetSubMessage(this IoBuffer buffer, ref SubMessage obj)
88          {
89              buffer.Align(4);
90              int smhPosition = buffer.Position;
91
92              SubMessageHeader header = buffer.GetSubMessageHeader();
93              int smStart = buffer.Position;
94
95              switch (header.SubMessageKind)
96              { // @see 9.4.5.1.1
97                  case SubMessageKind.PAD:
98                      Pad smPad = new Pad();
99                      smPad.Header = header;
100                     buffer.GetPad(ref smPad);
101                     obj = smPad;
102                     break;
103                 case SubMessageKind.ACKNACK:
104                     AckNack smAckNack = new AckNack();
105                     smAckNack.Header = header;
106                     buffer.GetAckNack(ref smAckNack);
107                     obj = smAckNack;
108                     break;
109                 case SubMessageKind.HEARTBEAT:
110                     Heartbeat smHeartbeat = new Heartbeat();
111                     smHeartbeat.Header = header;
112                     buffer.GetHeartbeat(ref smHeartbeat);
113                     obj = smHeartbeat;
114                     break;
115                 case SubMessageKind.GAP:
116                     Gap smgap = new Gap();
117                     smgap.Header = header;
118                     buffer.GetGap(ref smgap);
119                     obj = smgap;
120                     break;
121                 case SubMessageKind.INFO_TS:
122                     InfoTimestamp sminfots = new InfoTimestamp();
123                     sminfots.Header = header;
124                     buffer.GetInfoTimestamp(ref sminfots);
125                     obj = sminfots;
126                     break;
127                 case SubMessageKind.INFO_SRC:
128                     InfoSource smInfoSource = new InfoSource();
129                     smInfoSource.Header = header;
130                     buffer.GetInfoSource(ref smInfoSource);
131                     obj = smInfoSource;
132                     break;
133                 case SubMessageKind.INFO_REPLY_IP4:
134                     InfoReplyIp4 smInfoReplyIp4 = new InfoReplyIp4();
135                     smInfoReplyIp4.Header = header;
136                     buffer.GetInfoReplyIp4(ref smInfoReplyIp4);
137                     obj = smInfoReplyIp4;
138                     break;
139                 case SubMessageKind.INFO_DST:
140                     InfoDestination smInfoDestination = new InfoDestination();
141                     smInfoDestination.Header = header;
142                     buffer.GetInfoDestination(ref smInfoDestination);
143                     obj = smInfoDestination;
144                     break;
145                 case SubMessageKind.INFO_REPLY:
```

```
146                     InfoReply smInfoReply = new InfoReply();
147                     smInfoReply.Header = header;
148                     buffer.GetInfoReply(ref smInfoReply);
149                     obj = smInfoReply;
150                     break;
151                 case SubMessageKind.NACK_FRAG:
152                     NackFrag smNackFrag = new NackFrag();
153                     smNackFrag.Header = header;
154                     buffer.GetNackFrag(ref smNackFrag);
155                     obj = smNackFrag;
156                     break;
157                 case SubMessageKind.HEARTBEAT_FRAG:
158                     HeartbeatFrag smHeartbeatFrag = new HeartbeatFrag();
159                     smHeartbeatFrag.Header = header;
160                     buffer.GetHeartbeatFrag(ref smHeartbeatFrag);
161                     obj = smHeartbeatFrag;
162                     break;
163                 case SubMessageKind.DATA:
164                     Data smdata = new Data();
165                     smdata.Header = header;
166                     buffer.GetDataSubMessage(ref smdata);
167                     obj = smdata;
168                     break;
169                 case SubMessageKind.DATA_FRAG:
170                     DataFrag smdDataFrag = new DataFrag();
171                     smdDataFrag.Header = header;
172                     buffer.GetDataFrag(ref smdDataFrag);
173                     obj = smdDataFrag;
174                     break;
175 
176                 default:
177                     throw new NotSupportedException();
178             }
179 
180             int smEnd = buffer.Position;
181             int smLength = smEnd - smStart;
182             if (smLength != header.SubMessageLength && header.SubMessageLength != 0)
183             {
184                 log.WarnFormat("SubMessage length differs for {0} != {1} for {2}", smLength, header.
    SubMessageLength, obj);
185                 if (smLength < header.SubMessageLength)
186                 {
187                     byte[] unknownBytes = new byte[header.SubMessageLength - smLength];
188                     log.DebugFormat("Trying to skip {0} bytes", unknownBytes.Length);
189 
190                     buffer.Get(unknownBytes, 0, unknownBytes.Length);
191                 }
192             }
193 
194             log.DebugFormat("SubMsg in:  {0}", obj);
195         }
196     }
197 }
198 
```

```
1  using Mina.Core.Buffer;
2  using Rtps.Messages;
3  using Rtps.Messages.Types;
4
5  namespace Doopec.Rtps.Encoders
6  {
7      public static class SubMessageHeaderEncoder
8      {
9          public static void PutSubMessageHeader(this IoBuffer buffer, SubMessageHeader obj)
10         {
11             buffer.Put((byte)obj.SubMessageKind);
12             buffer.Put((byte)obj.FlagsValue);
13             buffer.PutInt16((short)obj.SubMessageLength);
14         }
15
16         public static SubMessageHeader GetSubMessageHeader(this IoBuffer buffer)
17         {
18             SubMessageHeader obj = new SubMessageHeader((SubMessageKind)0, 0);
19             buffer.GetSubMessageHeader(ref obj);
20             return obj;
21         }
22
23         public static void GetSubMessageHeader(this IoBuffer buffer, ref SubMessageHeader obj)
24         {
25             obj.SubMessageKind = (SubMessageKind)buffer.Get();
26             obj.FlagsValue = buffer.Get();
27             buffer.Order = (obj.IsLittleEndian ? ByteOrder.LittleEndian : ByteOrder.BigEndian); // Set the↙
       endianess
28             obj.SubMessageLength = (ushort)buffer.GetInt16();
29         }
30     }
31 }
32
```

```
1  using Mina.Core.Buffer;
2  using Rtps.Messages.Types;
3
4  namespace Doopec.Rtps.Encoders
5  {
6      public static class TimeEncoder
7      {
8          public static void PutTime(this IoBuffer buffer, Time obj)
9          {
10             buffer.PutInt32(obj.Seconds);
11             buffer.PutInt32((int)obj.Fraction);
12         }
13
14         public static Time GetTime(this IoBuffer buffer)
15         {
16             Time obj = new Time();
17             buffer.GetTime(ref obj);
18             return obj;
19         }
20
21         public static void GetTime(this IoBuffer buffer, ref Time obj)
22         {
23             obj.Seconds = buffer.GetInt32();
24             obj.Fraction = (uint)buffer.GetInt32(); ;
25         }
26     }
27 }
28
```

```
1  using Doopec.Serializer;
2  using Mina.Core.Buffer;
3  using Rtps.Structure.Types;
4  using System.Reflection;
5
6  namespace Doopec.Rtps.Encoders
7  {
8
9      public static class VendorIdEncoder
10     {
11         public static void PutVendorId(this IoBuffer buffer, VendorId obj)
12         {
13             buffer.Put(obj.ToBytes());
14         }
15         public static void WriteVendorId(IoBuffer buffer, VendorId obj)
16         {
17             buffer.Put(obj.ToBytes());
18         }
19         public static VendorId GetVendorId(this IoBuffer buffer)
20         {
21             VendorId obj = new VendorId();
22             buffer.GetVendorId(ref obj);
23             return obj;
24         }
25
26         public static void GetVendorId(this IoBuffer buffer, ref VendorId obj)
27         {
28             obj.Byte0 = buffer.Get();
29             obj.Byte1 = buffer.Get();
30         }
31
32         public static void ReadVendorId(IoBuffer buffer, ref VendorId obj)
33         {
34             if (obj == null)
35                 obj = new VendorId();
36             obj.Byte0 = buffer.Get();
37             obj.Byte1 = buffer.Get();
38         }
39     }
40     public class VendorIdSerializer : IStaticTypeSerializer
41     {
42         delegate void WriterDelegate(IoBuffer buffer, VendorId obj);
43         delegate void ReaderDelegate(IoBuffer buffer, ref VendorId obj);
44
45         public void GetStaticMethods(System.Type type, out MethodInfo writer, out MethodInfo reader)
46         {
47             WriterDelegate writerDelegate = VendorIdEncoder.WriteVendorId;
48             ReaderDelegate readerDelegate = VendorIdEncoder.ReadVendorId;
49             writer = writerDelegate.Method;
50             reader = readerDelegate.Method;
51         }
52
53         public bool Handles(System.Type type)
54         {
55             return type == typeof(VendorId);
56         }
57
58         public System.Collections.Generic.IEnumerable<System.Type> GetSubtypes(System.Type type)
59         {
60             yield break;
61         }
62     }
63 }
64
```

```
1  using log4net;
2  using Rtps.Structure;
3  using System;
4  using System.Collections.Generic;
5  using System.Linq;
6  using System.Reflection;
7  using System.Text;
8  using System.Threading.Tasks;
9
10 namespace Doopec.Rtps.SharedMem
11 {
12     public enum EventReason
13     {
14         NEW_PARTICIPANT,
15         DELETED_PARTICIPANT,
16         NEW_ENDPOINT,
17         DELETED_ENDPOINT,
18     }
19
20     public class DiscoveryEventArgs : EventArgs
21     {
22         public EventReason Reason { get; set; }
23         public object EventData { get; set; }
24     }
25
26     public class FakeDiscovery : IRtpsDiscovery
27     {
28         private static readonly ILog log = LogManager.GetLogger(MethodBase.GetCurrentMethod().
       DeclaringType);
29
30         private List<Participant> participants = new List<Participant>();
31         private List<Endpoint> endpoints = new List<Endpoint>();
32
33         public event DiscoveryEventHandler ParticipantDiscovery;
34         public event DiscoveryEventHandler EndpointDiscovery;
35
36         public IList<Participant> Participants
37         {
38             get { return participants.AsReadOnly(); }
39         }
40
41         public IList<Endpoint> Endpoints
42         {
43             get { return endpoints.AsReadOnly(); }
44         }
45
46
47
48         public void RegisterParticipant(Participant participant)
49         {
50             if (participant != null)
51             {
52                 participants.Add(participant);
53                 DiscoveryEventArgs dea = new DiscoveryEventArgs();
54                 dea.Reason = EventReason.NEW_PARTICIPANT;
55                 dea.EventData = participant;
56                 NotifyParticipantChanges(dea);
57             }
58         }
59
60         public void UnregisterParticipant(Participant participant)
61         {
62             if (participant != null)
63             {
64                 participants.Remove(participant);
65                 DiscoveryEventArgs dea = new DiscoveryEventArgs();
66                 dea.Reason = EventReason.DELETED_PARTICIPANT;
67                 dea.EventData = participant;
68                 NotifyParticipantChanges(dea);
69             }
70         }
71
72         public void RegisterEndpoint(Endpoint endpoint)
73         {
```

```
74              if (endpoint != null)
75              {
76                  endpoints.Add(endpoint);
77                  DiscoveryEventArgs dea = new DiscoveryEventArgs();
78                  dea.Reason = EventReason.NEW_ENDPOINT;
79                  dea.EventData = endpoint;
80                  NotifyEndpointsChanges(dea);
81              }
82          }
83
84          public void UnregisterEndpoint(Endpoint endpoint)
85          {
86              if (endpoint != null)
87              {
88                  endpoints.Remove(endpoint);
89                  DiscoveryEventArgs dea = new DiscoveryEventArgs();
90                  dea.Reason = EventReason.DELETED_ENDPOINT;
91                  dea.EventData = endpoint;
92                  NotifyEndpointsChanges(dea);
93              }
94          }
95
96          private void NotifyParticipantChanges(DiscoveryEventArgs dea)
97          {
98              log.Debug("The information about Participants has changed");
99              if (ParticipantDiscovery != null)
100             {
101                 ParticipantDiscovery(this, dea);
102             }
103         }
104
105         private void NotifyEndpointsChanges(DiscoveryEventArgs dea)
106         {
107             log.Debug("The information about Endpoints has changed");
108             if (EndpointDiscovery != null)
109             {
110                 EndpointDiscovery(this, dea);
111             }
112         }
113     }
114 }
115
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace Doopec.Rtps.SharedMem
8  {
9      internal class FakeEngine : IRtpsEngine
10     {
11         protected FakeDiscovery discoveryModule = new FakeDiscovery();
12
13
14         public IRtpsDiscovery DiscoveryModule
15         {
16             get { return discoveryModule; }
17         }
18     }
19 }
20
```

```
1
2  using Mina.Core.Buffer;
3  using Rtps.Messages.Submessages.Elements;
4  using System;
5  using Doopec.Rtps.Encoders;
6  using System.Diagnostics;
7
8  namespace Doopec.Rtps.Messages
9  {
10     /// <summary>
11     /// CDREncapsulation is a general purpose binary DataEncapsulation. It holds a IoBuffer,
12     /// which can be used by marshallers.
13     /// In addition to the encapsulation identifier, the OMG CDR encapsulation specifies the length of the data followed by the
14     /// data encapsulated using CDR. The same encapsulation scheme is used for both the length and serialized data.
15     /// 0...2...........8...............16..............24..............32
16     /// +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
17     /// |               CDR_BE          |         ushort options        |
18     /// +---------------+---------------+---------------+---------------+
19     /// |                   Data Length                                 |
20     /// +---------------+---------------+---------------+---------------+
21     /// ~               Serialized Data (CDR Big Endian)                ~
22     /// |                                                               |
23     /// +---------------+---------------+---------------+---------------+
24     ///
25     /// 0...2...........8...............16..............24..............32
26     /// +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
27     /// |               CDR_LE          |           ushort options      |
28     /// +---------------+---------------+---------------+---------------+
29     /// |                   Data Length                                 |
30     /// +---------------+---------------+---------------+---------------+
31     /// ~               Serialized Data (CDR Little Endian)             ~
32     /// |                                                               |
33     /// +---------------+---------------+---------------+---------------+
34     /// </summary>
35     public class CDREncapsulation : DataEncapsulation
36     {
37         private ByteOrder order;
38         private byte[] data;
39
40         public CDREncapsulation()
41         { }
42
43         public CDREncapsulation(IoBuffer buffer, object dataObj, ByteOrder order)
44         {
45             int initialPos = buffer.Position;
46             this.order = order;
47             buffer.Order = this.order;
48             if (order == ByteOrder.LittleEndian)
49                 buffer.PutEncapsulationScheme(CDR_LE_HEADER);
50             else
51                 buffer.PutEncapsulationScheme(CDR_BE_HEADER);
52
53             Doopec.Serializer.Serializer.Serialize(buffer, dataObj);
54             var serializedData = new byte[buffer.Position - initialPos];
55             buffer.Position = initialPos;
56             buffer.Get(serializedData, 0, serializedData.Length);
57             data = serializedData;
58         }
59
60         public static void Serialize(IoBuffer buffer, object dataObj, ByteOrder order)
61         {
62             buffer.Order = order;
63             if (order == ByteOrder.LittleEndian)
64                 buffer.PutEncapsulationScheme(CDR_LE_HEADER);
65             else
66                 buffer.PutEncapsulationScheme(CDR_BE_HEADER);
67             Doopec.Serializer.Serializer.Serialize(buffer, dataObj);
68         }
69
70         public static T Deserialize<T>(IoBuffer buffer)
71         {
72             EncapsulationScheme scheme = buffer.GetEncapsulationScheme();
```

```
73              if (scheme.Equals(DataEncapsulation.CDR_BE_HEADER))
74              {
75                  buffer.Order = ByteOrder.BigEndian;
76              }
77              else if (scheme.Equals(DataEncapsulation.CDR_LE_HEADER))
78              {
79                  buffer.Order = ByteOrder.LittleEndian;
80              }
81              else
82              {
83                  throw new NotImplementedException();
84              }
85              T rst = Doopec.Serializer.Serializer.Deserialize<T>(buffer);
86              return rst;
87          }
88
89          public static CDREncapsulation Deserialize(IoBuffer buffer, int length)
90          {
91              int initialPos = buffer.Position;
92              EncapsulationScheme scheme = buffer.GetEncapsulationScheme();
93              ByteOrder order;
94              if (scheme.Equals(DataEncapsulation.CDR_BE_HEADER))
95              {
96                  order = buffer.Order = ByteOrder.BigEndian;
97
98              }
99              else if (scheme.Equals(DataEncapsulation.CDR_LE_HEADER))
100             {
101                 order = buffer.Order = ByteOrder.LittleEndian;
102             }
103             else
104             {
105                 throw new NotImplementedException();
106             }
107             byte[] data = new byte[length - 4];
108             buffer.Get(data, 0, length - 4);
109             Debug.Assert(buffer.Position == initialPos + length);
110             CDREncapsulation rst = new CDREncapsulation(data, order);
111             return rst;
112         }
113
114         public CDREncapsulation(byte[] serializeData, ByteOrder order)
115         {
116             this.order = order;
117             data = serializeData;
118         }
119
120         public override byte[] SerializedPayload
121         {
122             get
123             {
124                 return data;
125             }
126         }
127
128         /// <summary>
129         /// Gets a IoBuffer, which can be used to marshall/unmarshall data.
130         /// </summary>
131         public IoBuffer Buffer
132         {
133             get
134             {
135                 IoBuffer buff = IoBuffer.Allocate(data.Length + 4);
136                 buff.Order = this.order;
137                 if (order == ByteOrder.LittleEndian)
138                     buff.PutEncapsulationScheme(CDR_LE_HEADER);
139                 else
140                     buff.PutEncapsulationScheme(CDR_BE_HEADER);
141                 buff.Put(data);
142                 return buff;
143             }
144         }
145
146         public override bool ContainsData()
```

```
147         {
148             return (data != null);
149         }
150     }
151 }
152
```

```
using Mina.Core.Buffer;
using Rtps.Messages.Submessages.Elements;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Doopec.Rtps.Encoders;
using Doopec.Serializer.Attributes;
using Doopec.Encoders;

namespace Doopec.Rtps.Messages
{
    public class EncapsulationManager
    {
        public static DataEncapsulation Serialize<T>(T obj, Encapsulation scheme = Encapsulation.CDR_BE)
        {
            DataEncapsulation rst;

            if (scheme == Encapsulation.UNKNOWN)
            {
                PacketAttribute packetAttr = PacketAttribute.GetAttribute(typeof(T));
                if (packetAttr != null)
                    scheme = packetAttr.EncapsulationScheme;
            }

             IoBuffer buff = IoBuffer.Allocate(1024);
             buff.AutoExpand = true;
             switch (scheme)
             {
                 default:
                 case Encapsulation.CDR_BE:
                     rst = buff.EncapsuleCDRData(obj, ByteOrder.BigEndian);
                     break;
                 case Encapsulation.CDR_LE:
                     rst = buff.EncapsuleCDRData(obj, ByteOrder.LittleEndian);
                     break;
                 case Encapsulation.PL_CDR_BE:
                     rst = buff.EncapsuleParameterListData(obj, ByteOrder.BigEndian);
                     break;
                 case Encapsulation.PL_CDR_LE:
                     rst = buff.EncapsuleParameterListData(obj, ByteOrder.LittleEndian);
                     break;
             }
             return rst;
        }

        public static DataEncapsulation Deserialize(IoBuffer buffer, int length)
        {
            int pos = buffer.Position;
            EncapsulationScheme scheme = buffer.GetEncapsulationScheme();
            buffer.Position = pos;
            if (scheme.Equals(DataEncapsulation.CDR_BE_HEADER) || scheme.Equals(DataEncapsulation.
    CDR_LE_HEADER))
            {
                return CDREncapsulation.Deserialize(buffer, length);
            }
            else if (scheme.Equals(DataEncapsulation.PL_CDR_BE_HEADER) || scheme.Equals(DataEncapsulation.
    PL_CDR_LE_HEADER))
            {
                return ParameterListEncapsulation.Deserialize(buffer, length);
            }
            else
                throw new ApplicationException("Unkonw scheme encapsulation " + scheme);

        }

        public static T Deserialize<T>(IoBuffer buffer) where T : new()
        {
            int pos = buffer.Position;
            EncapsulationScheme scheme = buffer.GetEncapsulationScheme();
            buffer.Position = pos;
            if (scheme.Equals(DataEncapsulation.CDR_BE_HEADER) || scheme.Equals(DataEncapsulation.
    CDR_LE_HEADER))
```

```
72              {
73                  return CDREncapsulation.Deserialize<T>(buffer);
74              }
75              else if (scheme.Equals(DataEncapsulation.PL_CDR_BE_HEADER) || scheme.Equals(DataEncapsulation.↙
    PL_CDR_LE_HEADER))
76              {
77                  return ParameterListEncapsulation.Deserialize<T>(buffer);
78              }
79              else
80                  throw new ApplicationException("Unkonw scheme encapsulation " + scheme);
81
82          }
83      }
84  }
85
```

```
using Doopec.Rtps.Encoders;
using Mina.Core.Buffer;
using Rtps.Messages.Submessages.Elements;
using System.Reflection;
using System;
using System.Linq;
using org.omg.dds.type;
using Rtps.Messages.Types;
using System.Diagnostics;
using Doopec.XTypes;
using org.omg.dds.type.typeobject;

namespace Doopec.Rtps.Messages
{
    /// <summary>
    /// ParameterListEncapsulation is a specialization of DataEncapsulation.
    /// In addition to the encapsulation identifier, the ParameterList encapsulation specifies the length of the data followed by the
    /// data encapsulated using a ParameterList. The same CDR encoding is used for both the length and the parameter list.
    /// 0...2...........8...............16..............24..............32
    /// +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    /// |             PL_CDR_BE         |         ushort options        |
    /// +---------------+---------------+---------------+---------------+
    /// |                                                               |
    /// ~        Serialized Data (ParameterList CDR Big Endian)         ~
    /// |                                                               |
    /// +---------------+---------------+---------------+---------------+
    /// 0...2...........8...............16..............24..............32
    /// +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    /// |             PL_CDR_LE         |         ushort options        |
    /// +---------------+---------------+---------------+---------------+
    /// |                                                               |
    /// ~        Serialized Data (ParameterList CDR Little Endian)      ~
    /// |                                                               |
    /// +---------------+---------------+---------------+---------------+
    ///
    /// And Serialized Data is:
    /// ....2...........8...............16..............24..............32
    /// +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    /// |       short parameterId_1     |       short length_1          |
    /// +---------------+---------------+---------------+---------------+
    /// |                                                               |
    /// ~                    octet[length_1] value_1                    ~
    /// |                                                               |
    /// +---------------+---------------+---------------+---------------+
    /// |       short parameterId_2     |      short length_2           |
    /// +---------------+---------------+---------------+---------------+
    /// |                                                               |
    /// ~                    octet[length_2] value_2                    ~
    /// |                                                               |
    /// +---------------+---------------+---------------+---------------+
    /// |                                                               |
    /// ~                            ...                                ~
    /// |                                                               |
    /// |                                                               |
    /// +---------------+---------------+---------------+---------------+
    /// |          PID_SENTINEL         |           ignored             |
    /// +---------------+---------------+---------------+---------------+
    ///
    /// This encapsulation is used by IsDiscovery.
    /// </summary>
    public class ParameterListEncapsulation : DataEncapsulation
    {
        private ParameterList parameters;
        private byte[] data;
        private ByteOrder order;

        public ParameterListEncapsulation(IoBuffer buffer, object dataObj, ByteOrder order)
        {
            int initialPos = buffer.Position;
            this.order = order;
            if (order == ByteOrder.LittleEndian)
                buffer.PutEncapsulationScheme(PL_CDR_LE_HEADER);
```

```
73              else
74                  buffer.PutEncapsulationScheme(PL_CDR_BE_HEADER);
75              buffer.Order = this.order;
76              ParameterList parameters = BuildParameters(dataObj, order);
77              buffer.PutParameterList(parameters);
78              data = new byte[buffer.Position - initialPos];
79              buffer.Position = initialPos;
80              buffer.Get(data, 0, data.Length);
81          }
82
83          internal ParameterListEncapsulation(IoBuffer buffer, ByteOrder order, int length)
84          {
85              this.order = order;
86              this.data = new byte[length];
87              buffer.Get(data, 0, data.Length);
88          }
89
90          public static void Serialize(IoBuffer buffer, object dataObj, ByteOrder order)
91          {
92              buffer.Order = order;
93              ParameterList parameters = BuildParameters(dataObj, order);
94              Serialize(buffer, parameters, order);
95          }
96          public static void Serialize(IoBuffer buffer, ParameterList parameters, ByteOrder order)
97          {
98              if (order == ByteOrder.LittleEndian)
99                  buffer.PutEncapsulationScheme(PL_CDR_LE_HEADER);
100             else
101                 buffer.PutEncapsulationScheme(PL_CDR_BE_HEADER);
102             buffer.Order = order;
103             int initialPos = buffer.Position;
104             buffer.Position += 4;
105             buffer.PutParameterList(parameters);
106             int finalPos = buffer.Position;
107             buffer.Position = initialPos;
108             buffer.PutInt32(finalPos - initialPos - 4);
109             buffer.Position = finalPos;
110         }
111
112         private static ParameterList BuildParameters(object obj, ByteOrder order)
113         {
114             var type = TypeExplorer.ExploreType(obj.GetType());
115             //var fields = obj.GetType().GetFields(BindingFlags.Public | BindingFlags.NonPublic |
    BindingFlags.Instance | BindingFlags.DeclaredOnly)
116             //                   .Where(fi => (fi.Attributes & FieldAttributes.NotSerialized) == 0);
117             StructureType structType = type as StructureType;
118             ParameterList parameters = new ParameterList();
119             foreach (var member in structType.GetMember())
120             {
121                 Parameter parameter = new Parameter();
122                 //uint id = field.GetProperty().MemberId;
123                 parameter.ParameterId = (ParameterId)member.GetProperty().MemberId; ;
124                 IoBuffer buffer = ByteBufferAllocator.Instance.Allocate(64);
125                 buffer.Order = order;
126                 buffer.AutoExpand = true;
127                 if (member.GetProperty().IsProperty)
128                 {
129                     var field = obj.GetType().GetProperty(member.GetProperty().Name);
130                     object data = field.GetValue(obj);
131                     if (data == null)
132                         continue;
133                     Doopec.Serializer.Serializer.Serialize(buffer, data);
134                 }
135                 else
136                 {
137                     var field = obj.GetType().GetField(member.GetProperty().Name);
138                     Doopec.Serializer.Serializer.Serialize(buffer, field.GetValue(obj));
139                 }
140
141                 int length = buffer.Position;
142                 parameter.Bytes = new byte[length];
143                 buffer.Rewind();
144                 buffer.Get(parameter.Bytes, 0, length);
145                 parameters.Value.Add(parameter);
```

```
146             }
147             return parameters;
148         }
149         public static T Deserialize<T>(IoBuffer buffer) where T : new()
150         {
151             EncapsulationScheme scheme = buffer.GetEncapsulationScheme();
152             if (scheme.Equals(DataEncapsulation.PL_CDR_BE_HEADER))
153             {
154                 buffer.Order = ByteOrder.BigEndian;
155             }
156             else if (scheme.Equals(DataEncapsulation.PL_CDR_LE_HEADER))
157             {
158                 buffer.Order = ByteOrder.LittleEndian;
159             }
160             else
161             {
162                 throw new NotImplementedException();
163             }
164             int initialPos = buffer.Position;
165             ParameterList parameters = buffer.GetParameterList();
166             return BuildObject<T>(parameters, buffer.Order);
167         }
168 
169         public static ParameterListEncapsulation Deserialize(IoBuffer buffer, int length)
170         {
171             int initialPos = buffer.Position;
172             EncapsulationScheme scheme = buffer.GetEncapsulationScheme();
173             if (scheme.Equals(DataEncapsulation.PL_CDR_BE_HEADER))
174             {
175                 buffer.Order = ByteOrder.BigEndian;
176             }
177             else if (scheme.Equals(DataEncapsulation.PL_CDR_LE_HEADER))
178             {
179                 buffer.Order = ByteOrder.LittleEndian;
180             }
181             else
182             {
183                 throw new NotImplementedException();
184             }
185             buffer.Position = initialPos;
186             return new ParameterListEncapsulation(buffer, buffer.Order, length);
187         }
188 
189         private static T BuildObject<T>(ParameterList parameters, ByteOrder order) where T : new()
190         {
191             var type = TypeExplorer.ExploreType(typeof(T));
192 
193             T obj = new T();
194             int cnt = 0;
195             StructureType structType = type as StructureType;
196             foreach (var member in structType.GetMember())
197             {
198                 Parameter parameter = parameters.Value.Where(p => (uint)p.ParameterId == member.
    GetProperty().MemberId).FirstOrDefault();
199                 if (parameter == null)
200                     continue;
201                 IoBuffer buffer = IoBuffer.Wrap(parameter.Bytes);
202                 buffer.Order = order;
203                 MethodInfo method = typeof(Doopec.Serializer.Serializer).GetMethods().Where(x => x.Name =
    = "Deserialize" && x.IsGenericMethod).SingleOrDefault(); ;
204                 if (member.GetProperty().IsProperty)
205                 {
206                     var field = obj.GetType().GetProperty(member.GetProperty().Name);
207 
208                     MethodInfo generic = method.MakeGenericMethod(field.PropertyType);
209                     object val = generic.Invoke(null, new object[] { buffer });
210                     field.SetValue(obj, val);
211                 }
212                 else
213                 {
214                     var field = obj.GetType().GetField(member.GetProperty().Name);
215 
216                     MethodInfo generic = method.MakeGenericMethod(field.FieldType);
217                     object val = generic.Invoke(null, new object[] { buffer });
```

```
218                     field.SetValue(obj, val);
219                 }
220             }
221             return obj;
222         }
223 
224         public ParameterListEncapsulation(ParameterList parameters, ByteOrder order)
225         {
226             this.parameters = parameters;
227             this.order = order;
228         }
229 
230         public ParameterList GetParameterList()
231         {
232             return parameters;
233         }
234 
235 
236         public override bool ContainsData()
237         {
238             return (data != null); // TODO: how do we represent key in serialized payload
239         }
240 
241         public override byte[] SerializedPayload
242         {
243             get
244             {
245                 return data;
246             }
247         }
248     }
249 }
250 
```

```
1  using Doopec.Rtps.SharedMem;
2  using log4net;
3  using Rtps.Structure;
4  using System;
5  using System.Collections.Generic;
6  using System.Linq;
7  using System.Reflection;
8  using System.Text;
9  using System.Threading.Tasks;
10
11 namespace Doopec.Rtps.RtpsTransport
12 {
13     public class RtpsDiscovery : IRtpsDiscovery
14     {
15         private static readonly ILog log = LogManager.GetLogger(MethodBase.GetCurrentMethod().
       DeclaringType);
16
17         private List<Participant> participants = new List<Participant>();
18         private List<Endpoint> endpoints = new List<Endpoint>();
19
20         public event DiscoveryEventHandler ParticipantDiscovery;
21
22         public event DiscoveryEventHandler EndpointDiscovery;
23
24
25
26         public IList<Participant> Participants
27         {
28             get { return participants.AsReadOnly(); }
29         }
30
31         public IList<Endpoint> Endpoints
32         {
33             get { return endpoints.AsReadOnly(); }
34         }
35
36         public void RegisterParticipant(Participant participant)
37         {
38             if (participant != null)
39             {
40                 participants.Add(participant);
41                 DiscoveryEventArgs dea = new DiscoveryEventArgs();
42                 dea.Reason = EventReason.NEW_PARTICIPANT;
43                 dea.EventData = participant;
44                 NotifyParticipantChanges(dea);
45             }
46         }
47
48         public void UnregisterParticipant(Participant participant)
49         {
50             if (participant != null)
51             {
52                 participants.Remove(participant);
53                 DiscoveryEventArgs dea = new DiscoveryEventArgs();
54                 dea.Reason = EventReason.DELETED_PARTICIPANT;
55                 dea.EventData = participant;
56                 NotifyParticipantChanges(dea);
57             }
58         }
59
60         public void RegisterEndpoint(Endpoint endpoint)
61         {
62             if (endpoint != null)
63             {
64                 endpoints.Add(endpoint);
65                 DiscoveryEventArgs dea = new DiscoveryEventArgs();
66                 dea.Reason = EventReason.NEW_ENDPOINT;
67                 dea.EventData = endpoint;
68                 NotifyEndpointsChanges(dea);
69             }
70         }
71
72         public void UnregisterEndpoint(Endpoint endpoint)
73         {
```

```
74              if (endpoint != null)
75              {
76                  endpoints.Remove(endpoint);
77                  DiscoveryEventArgs dea = new DiscoveryEventArgs();
78                  dea.Reason = EventReason.DELETED_ENDPOINT;
79                  dea.EventData = endpoint;
80                  NotifyEndpointsChanges(dea);
81              }
82          }
83
84          private void NotifyParticipantChanges(DiscoveryEventArgs dea)
85          {
86              log.Debug("The information about Participants has changed");
87              if (ParticipantDiscovery != null)
88              {
89                  ParticipantDiscovery(this, dea);
90              }
91          }
92
93          private void NotifyEndpointsChanges(DiscoveryEventArgs dea)
94          {
95              log.Debug("The information about Endpoints has changed");
96              if (EndpointDiscovery != null)
97              {
98                  EndpointDiscovery(this, dea);
99              }
100         }
101     }
102 }
103
```

```
1  using Doopec.Configuration;
2  using System;
3  using System.Collections.Generic;
4  using System.Configuration;
5
6  namespace Doopec.Rtps.RtpsTransport
7  {
8      public static class RtpsEngineFactory
9      {
10         private static IRtpsEngine theInstance;
11
12         public static IRtpsEngine Instance
13         {
14             get
15             {
16                 if (theInstance == null)
17                 { theInstance = RtpsEngineFactory.CreateEngine(null); }
18                 return theInstance;
19             }
20         }
21
22         public static IRtpsEngine CreateEngine(IDictionary<string, Object> environment)
23         {
24             DDSConfigurationSection ddsConfig = Doopec.Configuration.DDSConfigurationSection.Instance;
25             RTPSConfigurationSection rtpsConfig = Doopec.Configuration.RTPSConfigurationSection.Instance;
26             string transportProfile = ddsConfig.Domains[0].TransportProfile.Name;
27             string className = rtpsConfig.Transports[transportProfile].Type;
28             if (string.IsNullOrWhiteSpace(className))
29             {
30                 // no implementation class name specified
31                 throw new ApplicationException("Please Set the RTPS engine type property in the settings. ↙
       ");
32             }
33
34             Type ctxClass = Type.GetType(className, true);
35
36
37             // --- Instantiate new object --- //
38             try
39             {
40                 // First, try a constructor that will accept the environment.
41                 object newInstance = Activator.CreateInstance(ctxClass, environment);
42                 if (newInstance != null)
43                     return (IRtpsEngine)newInstance;
44             }
45             catch (Exception)
46             {
47                 /* No Map constructor found; try a no-argument constructor
48                  * instead.
49                  *
50                  * Get the constructor and call it explicitly rather than
51                  * calling Class.newInstance(). The latter propagates all
52                  * exceptions, even checked ones, complicating error handling
53                  * for us and the user.
54                  */
55                 object newInstance = Activator.CreateInstance(ctxClass);
56                 return (IRtpsEngine)newInstance;
57             }
58             throw new ApplicationException("Exception building a RTPS engine using " + className);
59         }
60     }
61
62     public class RtpsEngine : IRtpsEngine
63     {
64         protected RtpsDiscovery discoveryModule = new RtpsDiscovery();
65
66
67         public IRtpsDiscovery DiscoveryModule
68         {
69             get { return discoveryModule; }
70         }
71     }
72 }
73
```

```
using Doopec.Configuration;
using Doopec.Configuration.Dds;
using Doopec.Configuration.Rtps;
using Doopec.Rtps.Discovery;
using Doopec.Rtps.RtpsTransport;
using Rtps.Discovery.Spdp;
using Rtps.Structure;
using Rtps.Structure.Types;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Doopec.Rtps.Structure
{
    public class ParticipantImpl : Participant, IDisposable
    {
        private SPDPbuiltinParticipantReaderImpl spdpReader;

        //The SPDPbuiltinParticipantWriter is an RTPS Best-Effort StatelessWriter.
        private SPDPbuiltinParticipantWriterImpl spdpWriter;

        public int ParticipantId { get; set; }
        public int DomainId { get; set; }


        public ParticipantImpl(int domainId, int participantId)
            : base(new global::Rtps.Structure.Types.GUID(new GuidPrefix(domainId), EntityId.
    ENTITYID_PARTICIPANT))
        {
            this.DomainId = domainId;
            this.ParticipantId = participantId;
        }

        public void Enable()
        {
            DDSConfigurationSection ddsConfig = Doopec.Configuration.DDSConfigurationSection.Instance;
            RTPSConfigurationSection rtpsConfig = Doopec.Configuration.RTPSConfigurationSection.Instance;
            DomainParticipant domain = ddsConfig.Domains[this.DomainId];
            Transport transport = rtpsConfig.Transports[domain.TransportProfile.Name];

            spdpReader = new SPDPbuiltinParticipantReaderImpl(transport, this);
            spdpWriter = new SPDPbuiltinParticipantWriterImpl(transport, this);

            spdpReader.Start();
            spdpWriter.Start();
            RtpsEngineFactory.Instance.DiscoveryModule.RegisterParticipant(this);
        }

        public void Close()
        {
            //spdpReader.Close();
            //spdpWriter.Close();
            RtpsEngineFactory.Instance.DiscoveryModule.UnregisterParticipant(this);
            //throw new NotImplementedException();
        }

        public void Dispose()
        {
            Close();
        }
    }
}

```

```
1  using System.Configuration;
2
3  namespace Doopec.Rtps.Utils.Config
4  {
5      [ConfigurationCollection(typeof(KeyValueConfigurationElement), AddItemName = "Add",
6       CollectionType = ConfigurationElementCollectionType.BasicMap)]
7      public class ReaderConfig : KeyValueConfigurationCollection
8      {
9          public int HeartbeatResponseDelay
10         {
11             get { return int.Parse(this["heartbeatResponseDelay"].Value); }
12         }
13
14
15         public int HeartbeatSuppressionDuration
16         {
17             get { return int.Parse(this["heartbeatSuppressionDuration"].Value); }
18         }
19     }
20 }
21
```

```
1  using Doopec.Rtps.Utils.Config;
2  using System.Configuration;
3
4  namespace Doopec.Rtps.Config
5  {
6      /// <summary>
7      /// Class to register for the RTPS section of the configuration file
8      /// </summary>
9      /// <remarks>
10     /// The RTPS section of the configuration file needs to have a section
11     /// handler registered. This is the section handler used. It simply returns
12     /// the XML element that is the root of the section.
13     /// </remarks>
14     /// <example>
15     /// Example of registering the RTPS section handler :
16     /// <code lang="XML" escaped="true">
17     /// <configuration>
18     ///     <configSections>
19     ///         <section name="Doopec.Rtps" type="Doopec.Rtps.Config.RtpsConfigurationSectionHandler,
       Doopec" />
20     ///     </configSections>
21     ///     <Doopec.Rtps>
22     ///         RTPS configuration XML goes here
23     ///     </Doopec.Rtps>
24     /// </configuration>
25     /// </code>
26     /// </example>
27     public class RtpsConfigurationSectionHandler : ConfigurationSection
28     {
29         #region Public Instance Constructors
30
31         /// <summary>
32         /// Initializes a new instance of the <see cref="RtpsConfigurationSectionHandler"/> class.
33         /// </summary>
34         /// <remarks>
35         /// <para>
36         /// Default constructor.
37         /// </para>
38         /// </remarks>
39         public RtpsConfigurationSectionHandler()
40         {
41         }
42
43         #endregion Public Instance Constructors
44
45         [ConfigurationProperty("Doopec.Rtps.SPDP")]
46         public SpdpConfig SPDPConfig
47         {
48             get { return (SpdpConfig)this["Doopec.Rtps.SPDP"]; }
49         }
50
51         [ConfigurationProperty("Doopec.Rtps.Writer", IsDefaultCollection = true, IsKey = false, IsRequired
       = false)]
52         public WriterConfig WriterConfiguration
53         {
54             get { return (WriterConfig)this["Doopec.Rtps.Writer"]; }
55         }
56
57         [ConfigurationProperty("Doopec.Rtps.Reader", IsDefaultCollection = true, IsKey = false, IsRequired
       = false)]
58         public ReaderConfig ReaderConfiguration
59         {
60             get { return (ReaderConfig)this["Doopec.Rtps.Reader"]; }
61         }
62     }
63 }
64
```

```
1  using System.Configuration;
2
3  namespace Doopec.Rtps.Config
4  {
5      public class SpdpConfig : ConfigurationElement
6      {
7          private SpdpConfig() { }
8
9          [ConfigurationProperty("", IsDefaultCollection = true, IsKey = false, IsRequired = false)]
10         public SpdpSettingCollection SpdpSettings
11         {
12             get
13             {
14                 return  this[""] as SpdpSettingCollection;
15             }
16         }
17
18         public int ResendDataPeriod
19         {
20             get { return int.Parse(this.SpdpSettings["resendDataPeriod"].Value); }
21         }
22
23         public string UnicastLocatorList
24         {
25             get { return this.SpdpSettings["unicastLocatorList"].Value; }
26         }
27
28         public string MulticastLocatorList
29         {
30             get { return this.SpdpSettings["multicastLocatorList"].Value; }
31         }
32
33         [ConfigurationProperty("WellKnownPorts")]
34         public WellKnownPortsConfig WellKnownPorts
35         {
36             get { return (WellKnownPortsConfig)this["WellKnownPorts"]; }
37             set { this["WellKnownPorts"] = value; }
38         }
39
40     }
41
42
43     [ConfigurationCollection(typeof(KeyValueConfigurationElement), AddItemName = "Add",
44          CollectionType = ConfigurationElementCollectionType.BasicMap)]
45     public class SpdpSettingCollection : KeyValueConfigurationCollection
46    {
47
48    }
49 }
50
```

```
using System;
using System.Collections.Generic;
using System.Configuration;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Doopec.Rtps.Config
{
    public class WellKnownPortsConfig : ConfigurationElement
    {
        private WellKnownPortsConfig() { }

        /// <summary>
        /// Port Base number. This number sets the starting
        /// point for deriving port numbers used for Simple
        /// Endpoint Discovery Protocol (SEDP). This property
        /// is used in conjunction with DG, PG, D0 (or DX), and D1
        /// to construct the necessary Endpoints for RTPS
        /// IsDiscovery communication. (see section 9.6.1.1 in the
        /// OMG DDS-RTPS specification in how these
        /// Endpoints are constructed)
        /// </summary>
        [ConfigurationProperty("portBase", DefaultValue = "7400", IsRequired = true)]
        public int PortBase
        {
            get { return (int)this["portBase"]; }
            set { this["portBase"] = value; }
        }

        /// <summary>
        /// An integer Value representing the Domain Gain. This
        /// is a multiplier that assists in formulating Multicast
        /// or Unicast ports for RTPS.
        /// </summary>
        [ConfigurationProperty("domainIdGain", DefaultValue = "250", IsRequired = true)]
        public int DomainIdGain
        {
            get { return (int)this["domainIdGain"]; }
            set { this["domainIdGain"] = value; }
        }

        /// <summary>
        /// An integer that assists in configuring SPDP Unicast
        /// ports and serves as an offset multiplier as
        /// participants are assigned addresses using the
        /// formula:
        /// PB + DG * domainId + d1 + PG * participantId
        /// (see section 9.6.1.1 in the OMG DDS-RTPS
        /// specification in how these Endpoints are
        /// constructed)
        /// </summary>
        [ConfigurationProperty("participantIdGain", DefaultValue = "2", IsRequired = true)]
        public int ParticipantIdGain
        {
            get { return (int)this["participantIdGain"]; }
            set { this["participantIdGain"] = value; }
        }

        /// <summary>
        /// An integer Value that assists in providing an offset
        /// for calculating an assignable port in SPDP Multicast
        /// configurations. The formula used is:
        /// PB + DG * domainId + d0
        /// (see section 9.6.1.1 in the OMG DDS-RTPS
        /// specification in how these Endpoints are
        /// constructed)
        /// </summary>
        [ConfigurationProperty("offsetd0", DefaultValue = "0", IsRequired = true)]
        public int Offsetd0
        {
            get { return (int)this["offsetd0"]; }
            set { this["offsetd0"] = value; }
        }
```

```
75
76          /// <summary>
77          /// An integer Value that assists in providing an offset
78          /// for calculating an assignable port in SPDP Unicast
79          /// configurations. The formula used is:
80          /// PB + DG * domainId + d1 + PG * participantId
81          /// (see section 9.6.1.1 in the OMG DDS-RTPS
82          /// specification in how these Endpoints are
83          /// constructed)
84          /// </summary>
85          [ConfigurationProperty("offsetd1", DefaultValue = "10", IsRequired = true)]
86          public int Offsetd1
87          {
88              get { return (int)this["offsetd1"]; }
89              set { this["offsetd1"] = value; }
90          }
91
92          [ConfigurationProperty("offsetd2", DefaultValue = "1", IsRequired = true)]
93          public int Offsetd2
94          {
95              get { return (int)this["offsetd2"]; }
96              set { this["offsetd2"] = value; }
97          }
98
99          [ConfigurationProperty("offsetd3", DefaultValue = "11", IsRequired = true)]
100         public int Offsetd3
101         {
102             get { return (int)this["offsetd3"]; }
103             set { this["offsetd3"] = value; }
104         }
105     }
106 }
107
```

```
1  using System.Configuration;
2
3  namespace Doopec.Rtps.Utils.Config
4  {
5      [ConfigurationCollection(typeof(KeyValueConfigurationElement), AddItemName = "Add",
6       CollectionType = ConfigurationElementCollectionType.BasicMap)]
7      public class WriterConfig : KeyValueConfigurationCollection
8      {
9          public bool PushMode
10         {
11             get { return bool.Parse(this["pushMode"].Value); }
12         }
13
14         public int HeartbeatPeriod
15         {
16             get { return int.Parse(this["heartbeatPeriod"].Value); }
17         }
18
19
20         public int NackResponseDelay
21         {
22             get { return int.Parse(this["nackResponseDelay"].Value); }
23         }
24
25
26         public int NackSuppressionDuration
27         {
28             get { return int.Parse(this["nackSuppressionDuration"].Value); }
29         }
30     }
31 }
32
```

```
using Rtps.Structure.Types;
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Net.NetworkInformation;
using System.Text;
using System.Threading;
using System.Threading.Tasks;

namespace Doopec.Rtps.Utils
{
    /// <summary>
    /// Generate GuidPrefix_t values for use with RTPS
    ///    Also see GuidConverter.h in dds/DCPS
    ///    0  GUID_t.guidPrefix[ 0] == VendorId_t == 0x01 for OCI (used for OpenDDS)
    ///    1  GUID_t.guidPrefix[ 1] == VendorId_t == 0x03 for OCI (used for OpenDDS)
    ///    2  GUID_t.guidPrefix[ 2] == MAC Address
    ///    3  GUID_t.guidPrefix[ 3] == MAC Address
    ///    4  GUID_t.guidPrefix[ 4] == MAC Address
    ///    5  GUID_t.guidPrefix[ 5] == MAC Address
    ///    6  GUID_t.guidPrefix[ 6] == MAC Address
    ///    7  GUID_t.guidPrefix[ 7] == MAC Address
    ///    8  GUID_t.guidPrefix[ 8] == Process ID (MS byte)
    ///    9  GUID_t.guidPrefix[ 9] == Process ID (LS byte)
    ///   10  GUID_t.guidPrefix[10] == Counter (MS byte)
    ///   11  GUID_t.guidPrefix[11] == Counter (LS byte)
    ///
    /// Code borrowed from the code of OpenDDS
    /// </summary>
    public class GuidGenerator
    {
        /// <summary>
        /// Vendor Id Value specified for Doop-ec.
        /// TODO. Look for the right Value
        /// </summary>
        public static readonly byte[] VENDORID_DOOPEC = new byte[] { 0x01, 0xAA };

        public const int NODE_ID_SIZE = 6;

        private static Random rand = new Random();

        /// Static constructor - initializes pid and MAC address values
        static GuidGenerator()
        {
            nodeId[0] = VENDORID_DOOPEC[0];
            nodeId[1] = VENDORID_DOOPEC[1];

            byte[] partId = new byte[NODE_ID_SIZE];
            PhysicalAddress macaddress = GetMacAddress();

            if (macaddress != null)
            {
                Array.Copy(macaddress.GetAddressBytes(), partId, NODE_ID_SIZE);
            }
            else
            {
                rand.NextBytes(partId);
            }
            Array.Copy(partId, 0, nodeId, 2, NODE_ID_SIZE);

            int pid = Process.GetCurrentProcess().Id;
            nodeId[8] = (byte)(pid >> 8);
            nodeId[9] = (byte)(pid & 0xFF);
        }

        /// <summary>
        /// Finds the MAC address of the first operation NIC found.
        /// </summary>
        /// <returns>The MAC address.</returns>
        private static PhysicalAddress GetMacAddress()
        {
            foreach (NetworkInterface nic in NetworkInterface.GetAllNetworkInterfaces())
            {
```

```
75                  if (nic.OperationalStatus == OperationalStatus.Up)
76                  {
77                      return nic.GetPhysicalAddress();
78                  }
79              }
80              return null;
81          }
82
83          /// populate a GUID container with a unique ID. This will increment
84          /// the counter, and use a lock while doing so.
85          public void Populate(ref GUID container)
86          {
87              byte[] guidPrefix = container.Prefix.Prefix; // new byte[GuidPrefix.GUID_PREFIX_SIZE];
88              Array.Copy(nodeId, 0, guidPrefix, 0, nodeId.Length);
89
90              int count = GetCount();
91              guidPrefix[10] = (byte)(count >> 8);
92              guidPrefix[11] = (byte)(count & 0xFF);
93          }
94
95          public GUID GenerateGuid()
96          {
97              GUID container = new GUID();
98              Populate(ref container);
99              return container;
100         }
101
102
103         private static int GetCount()
104         {
105             return Interlocked.Increment(ref counter);
106         }
107
108         private static byte[] nodeId = new byte[NODE_ID_SIZE + 2 + 2];
109         private static int counter;
110     }
111 }
112
```

```
1  using System;
2  using System.Threading;
3  using System.Threading.Tasks;
4
5  namespace Doopec.Rtps.Utils
6  {
7      public class PeriodicWorker
8      {
9          // create the cancellation token source
10         private CancellationTokenSource tokenSource = new CancellationTokenSource();
11         private bool isRunning = false;
12
13         public int Period { get; set; }
14
15         public int Count { get; set; }
16
17         public void Start(int period)
18         {
19             this.Period = period;
20             isRunning = true;
21             KeepWorkerRunning();
22         }
23
24         public virtual void End()
25         {
26             isRunning = false;
27             // cancel the token
28             tokenSource.Cancel();
29         }
30
31         public bool IsRunning
32         { get { return isRunning; } }
33
34         public virtual void DoPeriodicWork()
35         {
36         }
37
38         private void KeepWorkerRunning()
39         {
40             // create the cancellation token
41             CancellationToken token = tokenSource.Token;
42
43             // create the first task, which we will let run fully
44             Task task = new Task(() =>
45             {
46                 while (isRunning)
47                 {
48                     // Doing some periodic work
49                     DoPeriodicWork();
50                     // put the task to sleep for this.Period milliseconds
51                     bool cancelled = token.WaitHandle.WaitOne(this.Period);
52                     Count++;
53                     // check to see if we have been cancelled
54                     if (cancelled)
55                     {
56                         throw new OperationCanceledException(token);
57                     }
58                 }
59             }, token);
60             // start task
61             task.Start();
62         }
63     }
64 }
65
```

```
1
2  using Doopec.Rtps.SharedMem;
3  using Rtps.Structure;
4  using System.Collections.Generic;
5  namespace Doopec.Rtps
6  {
7
8      public delegate void DiscoveryEventHandler(object sender, DiscoveryEventArgs e);
9
10     public interface IRtpsDiscovery
11     {
12         event DiscoveryEventHandler ParticipantDiscovery;
13         event DiscoveryEventHandler EndpointDiscovery;
14
15         void RegisterParticipant(Participant participant);
16
17         void UnregisterParticipant(Participant participant);
18
19         void RegisterEndpoint(Endpoint endpoint);
20
21         void UnregisterEndpoint(Endpoint endpoint);
22
23         IList<Participant> Participants { get; }
24
25         IList<Endpoint> Endpoints { get; }
26
27     }
28 }
29
```

```
1
2 namespace Doopec.Rtps
3 {
4     public interface IRtpsEngine
5     {
6         IRtpsDiscovery DiscoveryModule { get; }
7     }
8 }
9
```

```
1  using Rtps.Structure;
2  using System;
3  using System.Collections.Generic;
4  using System.Linq;
5  using System.Text;
6  using System.Threading.Tasks;
7
8  namespace Doopec.RTPS
9  {
10     /// <summary>
11     /// WriterCache represents writers history cache from the RTPSWriter point of
12     /// view. RTPSWriter uses WriterCache to construct Data and HeartBeat messages to
13     /// be sent to RTPSReaders
14     /// </summary>
15     /// <typeparam name="T"></typeparam>
16     public interface WriterCache<T>
17     {
18         /// <summary>
19         /// Gets the smallest sequence number available in history cache.
20         /// </summary>
21         /// <returns></returns>
22         long getSeqNumMin();
23
24         /// <summary>
25         /// Gets the greatest sequence number available in history cache.
26         /// </summary>
27         /// <returns>seqNumMax</returns>
28         long getSeqNumMax();
29
30         /// <summary>
31         /// Gets all the CacheChanges since given sequence number.
32         /// Returned CacheChanges are ordered by sequence numbers.
33         /// </summary>
34         /// <param name="seqNum">sequence number to compare</param>
35         /// <returns>changes since given seqNum. Returned List is newly allocated.</returns>
36         LinkedList<CacheChange<T>> getSamplesSince(long seqNum);
37
38     }
39 }
40
```

```
1  using Rtps.Structure.Types;
2
3  namespace Doopec.Utils.Transport
4  {
5      /// <summary>
6      /// Receiver will be used to receive packets from the source. Typically, source is from the network, ↙
       but
7      /// it can be anything. Like memory, file etc.
8      /// </summary>
9      public interface IReceiver
10     {
11         /// <summary>
12         /// Gets the locator associated with this Receiver. This locator will be transmitted
13         /// to remote participants.
14         /// </summary>
15         /// <returns></returns>
16         Locator Locator { get; }
17
18         /// <summary>
19         /// Gets the participantId associated with this receiver. During creation of receiver,
20         /// participantId may be given as -1, indicating that provider should generate one.
21         /// This method returns the Value assigned by the provider.
22         /// </summary>
23         GUID ParticipantId { get; }
24
25         void Start();
26
27         /// <summary>
28         /// Close this Receiver
29         /// </summary>
30         void Close();
31     }
32 }
33
```

```
1  using Rtps.Messages;
2  using Rtps.Structure.Types;
3
4  namespace Doopec.Utils.Transport
5  {
6
7      /// <summary>
8      /// Transmitter is used to deliver messages to destination.
9      /// </summary>
10     public interface ITransmitter
11     {
12         /// <summary>
13         /// Gets the locator associated with this Receiver. This locator will be transmitted
14         /// to remote participants.
15         /// </summary>
16         /// <returns></returns>
17         Locator Locator { get; }
18
19         /// <summary>
20         /// Gets the participantId associated with this receiver. During creation of receiver,
21         /// participantId may be given as -1, indicating that provider should generate one.
22         /// This method returns the Value assigned by the provider.
23         /// </summary>
24         GUID ParticipantId { get; }
25
26         /// <summary>
27         /// Sends a Message to destination.
28         /// </summary>
29         /// <param name="msg">Message to send</param>
30         /// <returns>true, if an overflow occured.</returns>
31         void SendMessage(Message msg);
32
33         void Start();
34
35         /// <summary>
36         /// Close this Writer
37         /// </summary>
38         void Close();
39     }
40 }
41
```

```
1  using Doopec.Rtps.Encoders;
2  using log4net;
3  using Mina.Core.Session;
4  using Mina.Filter.Codec;
5  using Mina.Transport.Socket;
6  using Rtps.Messages;
7  using Rtps.Structure.Types;
8  using System;
9  using System.Net;
10 using System.Net.Sockets;
11 using System.Reflection;
12
13 namespace Doopec.Utils.Transport
14 {
15     /// <summary>
16     /// Provides data for <see cref="IoSession"/>'s message receive/sent event.
17     /// </summary>
18     public class RTPSMessageEventArgs : IoSessionEventArgs
19     {
20         private readonly Message _message;
21
22         /// <summary>
23         /// </summary>
24         public RTPSMessageEventArgs(IoSession session, Message message)
25             : base(session)
26         {
27             _message = message;
28         }
29
30         /// <summary>
31         /// Gets the associated message.
32         /// </summary>
33         public Message Message
34         {
35             get { return _message; }
36         }
37     }
38
39     /// <summary>
40     /// This class receives UDP packets from the network.
41     /// </summary>
42     public class UDPReceiver : IReceiver, IDisposable
43     {
44         private static readonly ILog log = LogManager.GetLogger(MethodBase.GetCurrentMethod().
       DeclaringType);
45
46         private readonly int bufferSize;
47
48         private readonly Locator locator;
49         private GUID participantId;
50         private AsyncDatagramAcceptor acceptor;
51
52         public event EventHandler<RTPSMessageEventArgs> MessageReceived;
53         public bool IsDiscovery { get; set; }
54
55         public UDPReceiver(Uri uri, int bufferSize)
56         {
57             this.bufferSize = bufferSize;
58             var addresses = System.Net.Dns.GetHostAddresses(uri.Host);
59             int port = (uri.Port < 0 ? 0 : uri.Port);
60             if (addresses != null && addresses.Length >= 1)
61                 this.locator = new Locator(addresses[0], port);
62         }
63
64         public UDPReceiver(Locator locator, int bufferSize)
65         {
66             this.bufferSize = bufferSize;
67             this.locator = locator;
68         }
69
70         public Locator Locator
71         {
72             get { return locator; }
73         }
```

```
74
75          public GUID ParticipantId
76          {
77              get { return participantId; }
78              set { participantId = value; }
79          }
80
81          public void Start()
82          {
83              if (locator == null)
84                  throw new ApplicationException();
85
86              IPEndPoint ep = new IPEndPoint(locator.SocketAddress, locator.Port);
87              bool isMultiCastAddr;
88              if (ep.AddressFamily == AddressFamily.InterNetwork) //IP v4
89              {
90                  byte byteIp = ep.Address.GetAddressBytes()[0];
91                  isMultiCastAddr = (byteIp >= 224 && byteIp < 240) ? true : false;
92              }
93              else if (ep.AddressFamily == AddressFamily.InterNetworkV6)
94              {
95                  isMultiCastAddr = ep.Address.IsIPv6Multicast;
96              }
97              else
98              {
99                  throw new NotImplementedException("Address family not supported yet: " + ep.
    AddressFamily);
100             }
101             if (isMultiCastAddr)
102             {
103                 acceptor = new AsyncDatagramAcceptor();
104                 // Define a MulticastOption object specifying the multicast group
105                 // address and the local IPAddress.
106                 // The multicast group address is the same as the address used by the client.
107                 MulticastOption mcastOption = new MulticastOption(locator.SocketAddress, IPAddress.Any);
108                 acceptor.SessionConfig.MulticastOption = mcastOption;
109                 acceptor.SessionConfig.ExclusiveAddressUse = false;
110                 acceptor.SessionConfig.ReuseAddress = true;
111             }
112             else
113                 acceptor = new AsyncDatagramAcceptor();
114
115             //acceptor.FilterChain.AddLast("logger", new LoggingFilter());
116             acceptor.FilterChain.AddLast("RTPS", new ProtocolCodecFilter(new MessageCodecFactory()));
117             acceptor.SessionConfig.EnableBroadcast = true;
118
119             acceptor.ExceptionCaught += (s, e) =>
120             {
121                 Console.WriteLine(e.Exception);
122                 e.Session.Close(true);
123             };
124             acceptor.MessageReceived += (s, e) =>
125             {
126                 Message msg = e.Message as Message;
127                 if (MessageReceived != null)
128                     MessageReceived(s, new RTPSMessageEventArgs(e.Session, msg));
129                 //if (log.IsDebugEnabled)
130                 //{
131                 //    log.DebugFormat("New Message has arrived from {0}", e.Session.RemoteEndPoint);
132                 //    log.DebugFormat("Message Header: {0}", msg.Header);
133                 //    foreach (var submsg in msg.SubMessages)
134                 //    {
135                 //        log.DebugFormat("SubMessage: {0}", submsg);
136                 //        if (submsg is Data)
137                 //        {
138                 //            Data d = submsg as Data;
139                 //            foreach (var par in d.InlineQos.Value)
140                 //                log.DebugFormat("InlineQos: {0}", par);
141                 //        }
142                 //    }
143                 //}
144             };
145             acceptor.SessionCreated += (s, e) =>
146             {
```

```
147                 log.Debug("Session created...");
148             };
149             acceptor.SessionOpened += (s, e) =>
150             {
151                 log.Debug("Session opened...");
152             };
153             acceptor.SessionClosed += (s, e) =>
154             {
155                 log.Debug("Session closed...");
156             };
157             acceptor.SessionIdle += (s, e) =>
158             {
159                 log.Debug("Session idle...");
160             };
161             if (isMultiCastAddr)
162                 acceptor.Bind(new IPEndPoint(IPAddress.Any, locator.Port));
163             else
164                 acceptor.Bind(new IPEndPoint(locator.SocketAddress, locator.Port));
165             log.DebugFormat("Listening on udp://{0}:{1} for {2}", locator.SocketAddress, locator.Port,
    IsDiscovery ? "IsDiscovery traffic" : "user traffic");
166         }
167 
168         public void Close()
169         {
170             if (acceptor != null)
171             {
172                 log.DebugFormat("Closing {0}", acceptor.LocalEndPoint);
173 
174                 acceptor.Dispose();
175             }
176         }
177 
178         public void Dispose()
179         {
180             if (acceptor != null)
181                 acceptor.Dispose();
182         }
183     }
184 }
185 
```

```
using Doopec.Rtps.Encoders;
using log4net;
using Mina.Core.Future;
using Mina.Core.Service;
using Mina.Core.Session;
using Mina.Filter.Codec;
using Mina.Transport.Socket;
using Rtps.Messages;
using Rtps.Structure.Types;
using System;
using System.Net;
using System.Net.Sockets;
using System.Reflection;

namespace Doopec.Utils.Transport
{
    public class UDPTransmitter : ITransmitter, IDisposable
    {
        private static readonly ILog log = LogManager.GetLogger(MethodBase.GetCurrentMethod().
    DeclaringType);

        private readonly Locator locator;
        private GUID participantId;
        private AsyncDatagramConnector connector;
        private int bufferSize;
        private IoSession session;

        public bool IsDiscovery { get; set; }

        public Locator Locator
        {
            get { return locator; }
        }

        public GUID ParticipantId
        {
            get { return participantId; }
            set { participantId = value; }
        }


        public UDPTransmitter(Uri uri, int bufferSize)
        {
            var addresses = System.Net.Dns.GetHostAddresses(uri.Host);
            int port = (uri.Port < 0 ? 0 : uri.Port);
            if (addresses != null && addresses.Length >= 1)
                this.locator = new Locator(addresses[0], port);
        }

        /// <summary>
        /// Constructor for UDPTransmitter.
        /// </summary>
        /// <param name="locator">Locator where the messages will be sent.</param>
        /// <param name="bufferSize">Size of the buffer that will be used to Write messages.</param>
        public UDPTransmitter(Locator locator, int bufferSize)
        {
            this.locator = locator;
            this.bufferSize = bufferSize;
        }

        public void Start()
        {
            IPEndPoint ep = new IPEndPoint(locator.SocketAddress, locator.Port);
            bool isMultiCastAddr;
            if (ep.AddressFamily == AddressFamily.InterNetwork) //IP v4
            {
                byte byteIp = ep.Address.GetAddressBytes()[0];
                isMultiCastAddr = (byteIp >= 224 && byteIp < 240) ? true : false;
            }
            else if (ep.AddressFamily == AddressFamily.InterNetworkV6)
            {
                isMultiCastAddr = ep.Address.IsIPv6Multicast;
            }
            else
```

```
74              {
75                  throw new NotImplementedException("Address family not supported yet: " + ep.
    AddressFamily);
76              }
77
78              connector = new AsyncDatagramConnector();
79
80              connector.FilterChain.AddLast("RTPS", new ProtocolCodecFilter(new MessageCodecFactory()));
81
82              if (isMultiCastAddr)
83              {
84                  // Set the local IP address used by the listener and the sender to
85                  // exchange multicast messages.
86                  connector.DefaultLocalEndPoint = new IPEndPoint(IPAddress.Any, 0);
87
88                  // Define a MulticastOption object specifying the multicast group
89                  // address and the local IP address.
90                  // The multicast group address is the same as the address used by the listener.
91                  MulticastOption mcastOption = new MulticastOption(locator.SocketAddress, IPAddress.Any);
92                  connector.SessionConfig.MulticastOption = mcastOption;
93
94                  // Call Connect() to force binding to the local IP address,
95                  // and get the associated multicast session.
96                  IoSession session = connector.Connect(ep).Await().Session;
97              }
98
99              connector.ExceptionCaught += (s, e) =>
100             {
101                 log.Error(e.Exception);
102             };
103             connector.MessageReceived += (s, e) =>
104             {
105                 log.Debug("Session recv...");
106             };
107             connector.MessageSent += (s, e) =>
108             {
109                 log.Debug("Session sent...");
110             };
111             connector.SessionCreated += (s, e) =>
112             {
113                 log.Debug("Session created...");
114             };
115             connector.SessionOpened += (s, e) =>
116             {
117                 log.Debug("Session opened...");
118             };
119             connector.SessionClosed += (s, e) =>
120             {
121                 log.Debug("Session closed...");
122             };
123             connector.SessionIdle += (s, e) =>
124             {
125                 log.Debug("Session idle...");
126             };
127             IConnectFuture connFuture = connector.Connect(new IPEndPoint(locator.SocketAddress, locator.
    Port));
128             connFuture.Await();
129
130             connFuture.Complete += (s, e) =>
131             {
132                 IConnectFuture f = (IConnectFuture)e.Future;
133                 if (f.Connected)
134                 {
135                     log.Debug("...connected");
136                     session = f.Session;
137                 }
138                 else
139                 {
140                     log.Warn("Not connected...exiting");
141                 }
142             };
143         }
144
145         /**
```

```
146          * Sends a Message to a Locator of this UDPWriter.
147          * If an overflow occurs during writing of Message, only submessages that
148          * were succesfully written will be sent.
149          *
150          * @param m Message to send
151          * @return true, if Message did not fully fit into buffer of this UDPWriter
152          */
153         public void SendMessage(Message m)
154         {
155             if (session != null)
156             {
157                 session.Write(m);
158             }
159         }
160 
161         public void Close()
162         {
163             session.Close(false);
164         }
165 
166 
167 
168         public void Dispose()
169         {
170             this.Close();
171         }
172     }
173 }
174 
```

```
1  using Doopec.Dds.XTypes;
2  using org.omg.dds.type;
3  using org.omg.dds.type.typeobject;
4  using Rtps.Attributes;
5  using System;
6  using System.Collections.Generic;
7  using System.Diagnostics;
8  using System.Reflection;
9
10 namespace Doopec.XTypes
11 {
12     public static class TypeExplorer
13     {
14         public static org.omg.dds.type.typeobject.Type ExploreType(System.Type type)
15         {
16             org.omg.dds.type.typeobject.Type ddsType = null;
17             if (type.IsClass || type.IsValueType)
18             {
19                 ddsType = ExploreClass(type);
20             }
21             else if (type.IsEnum)
22             {
23                 ddsType = ExploreEnum(type);
24             }
25             else
26             {
27                 throw new NotImplementedException();
28             }
29             return ddsType;
30         }
31
32         private static StructureType ExploreClass(System.Type type)
33         {
34             StructureType ddsType = new StructureTypeImpl();
35
36             TypeProperty typeProp = new TypePropertyImpl();
37             typeProp.SetName(type.FullName);
38             typeProp.SetTypeId(type.GetHashCode());
39             TypeFlag flag = default(TypeFlag);
40             flag |= (type.IsSealed ? TypeFlag.IS_FINAL : 0);
41             typeProp.SetFlag(flag);
42             ddsType.SetProperty(typeProp);
43
44             List<Member> listMember = new List<Member>();
45             var fields = type.GetFields(BindingFlags.Public |
46                                         BindingFlags.Instance);
47             uint lastId = 0;
48             foreach (var member in fields)
49             {
50                 Member memberInfo = new MemberImpl();
51                 NonFieldAttribute isField = member.GetCustomAttribute<NonFieldAttribute>();
52                 if (isField != null)
53                     continue;
54
55                 MemberProperty memberProp = new MemberPropertyImpl();
56                 memberProp.SetName(member.Name);
57                 IDAttribute id = member.GetCustomAttribute<IDAttribute>();
58                 if (id != null)
59                 {
60                     memberProp.SetMemberId(id.Value);
61                     lastId = id.Value;
62                 }
63                 else
64                 {
65                     lastId++;
66                     memberProp.SetMemberId(lastId);
67                 }
68                 MemberFlag memberFlag = default(MemberFlag);
69                 KeyAttribute isKey = member.GetCustomAttribute<KeyAttribute>();
70                 if (isKey != null)
71                 {
72                     memberFlag |= MemberFlag.IS_KEY;
73                 }
74                 OptionalAttribute isOptional = member.GetCustomAttribute<OptionalAttribute>();
```

```
75                  if (isOptional != null)
76                  {
77                      memberFlag |= MemberFlag.IS_OPTIONAL;
78                  }
79
80                  memberProp.SetFlag(memberFlag);
81                  memberProp.IsProperty = false;
82                  memberInfo.SetProperty(memberProp);
83                  listMember.Add(memberInfo);
84              }
85              var props = type.GetProperties(BindingFlags.Public |
86                                             BindingFlags.Instance);
87              foreach (var member in props)
88              {
89                  Member memberInfo = new MemberImpl();
90                  NonFieldAttribute isField = member.GetCustomAttribute<NonFieldAttribute>();
91                  if (isField != null)
92                      continue;
93                  MemberProperty memberProp = new MemberPropertyImpl();
94                  memberProp.SetName(member.Name);
95                  IDAttribute id = member.GetCustomAttribute<IDAttribute>();
96                  if (id != null)
97                  {
98                      memberProp.SetMemberId(id.Value);
99                      lastId = id.Value;
100                 }
101                 else
102                 {
103                     continue; // TODO. Check if only process process members with ID
104                     //lastId++;
105                     //memberProp.SetMemberId(lastId);
106                 }
107                 MemberFlag memberFlag = default(MemberFlag);
108                 KeyAttribute isKey = member.GetCustomAttribute<KeyAttribute>();
109                 if (isKey != null)
110                 {
111                     memberFlag |= MemberFlag.IS_KEY;
112                 }
113                 OptionalAttribute isOptional = member.GetCustomAttribute<OptionalAttribute>();
114                 if (isOptional != null)
115                 {
116                     memberFlag |= MemberFlag.IS_OPTIONAL;
117                 }
118
119                 memberProp.SetFlag(memberFlag);
120                 memberProp.IsProperty = true;
121                 memberInfo.SetProperty(memberProp);
122                 listMember.Add(memberInfo);
123             }
124             ddsType.SetMember(listMember);
125
126             return ddsType;
127         }
128
129         private static EnumerationType ExploreEnum(System.Type type)
130         {
131             EnumerationType ddsType = new EnumerationTypeImpl();
132
133             TypeProperty typeProp = new TypePropertyImpl();
134             typeProp.SetName(type.FullName);
135             typeProp.SetTypeId(type.GetHashCode());
136             TypeFlag flag = default(TypeFlag);
137             //TODO flag |= (type.IsSealed? TypeFlag.IS_FINAL : 0);
138             typeProp.SetFlag(flag);
139             ddsType.SetProperty(typeProp);
140
141             var bitbound = type.GetCustomAttribute<BitBoundAttribute>();
142             if (bitbound != null)
143             {
144                 Debug.Assert(bitbound.Value >= 1 && bitbound.Value <= 32, "The Value member may Take any ↙
    Value from 1 to 32, inclusive, when this annotation is applied to an enumerated type.");
145
146                 ddsType.SetBitBound(bitbound.Value);
147             }
```

```
148             else
149             {
150                 var ut = type.GetEnumUnderlyingType();
151                 if (ut == typeof(int) || ut == typeof(uint))
152                 {
153                     ddsType.SetBitBound(32);
154                 }
155                 else if (ut == typeof(byte) || ut == typeof(sbyte))
156                 {
157                     ddsType.SetBitBound(8);
158                 }
159                 else if (ut == typeof(short) || ut == typeof(ushort))
160                 {
161                     ddsType.SetBitBound(16);
162                 }
163                 else if (ut == typeof(long) || ut == typeof(ulong))
164                 {
165                     ddsType.SetBitBound(64);
166                 }
167                 else { throw new NotSupportedException(ut.ToString()); }
168             }
169             IList<EnumeratedConstant> listConstants = new List<EnumeratedConstant>();
170             var fields = type.GetFields();
171             var constantNames = type.GetEnumNames();
172             var constantValues = type.GetEnumValues();
173             for (int i = 0; i < constantNames.Length; i++)
174             {
175                 EnumeratedConstant constant = new EnumeratedConstantImpl();
176                 constant.SetName(constantNames[i]);
177                 constant.SetValue((int)constantValues.GetValue(i));
178                 listConstants.Add(constant);
179             }
180             ddsType.SetConstant(listConstants);
181             return ddsType;
182         }
183     }
184 }
185
```