# CAPÍTULO IV

**PRUEBAS**

## INTRODUCCIÓN

En este capítulo primeramente se realizarán pruebas unitarias para la comprobación de distintos componentes del programa tales como clases y métodos, las cuáles estarán documentadas mediante una estructura de tablas donde se incluye la llamada, la descripción, la entrada, la referencia, el código y la salida.

A continuación se describirá el ambiente de pruebas, el cual tendrá como base a 4 computadores comunicándose entre si por medio del protocolo RTPS y en contraste se realizará la misma prueba con el protocolo RT-CORBA o Ada-DSA o DRTSJ. Además se adjunta capturas de pantalla tanto de la aplicación utilizando los protocolos y de capturas del flujo de datos con la herramienta wireshark, y se presenta un manual de usuario de las aplicaciones y del protocolo RTPS.

Finalmente, se realizará una comparación midiendo tiempos de respuesta y eficiencia del protocolo dentro de nuestro ambiente de pruebas.

## Pruebas unitarias del api rtps

### Codificadores

#### Prueba de los Elementos de los Mensajes.

|  |  |
| --- | --- |
| **Llamada:**  public static DataEncapsulation Serialize<T>(T obj, Encapsulation scheme = Encapsulation.CDR\_BE)  public static DataEncapsulation Deserialize(IoBuffer buffer, int length) | |
| ***Descripción*** | En esta prueba se verifica que el fichero de configuración no sea nulo |
| ***Entrada*** | Inicialmente se tiene el fichero de configuración |
| ***Referencia*** |  |
| ***Código*** | [TestMethod]  public void TestLocatorIpV4CDR\_BE()  {  Encapsulation Scheme = Encapsulation.CDR\_BE;  int bufferSize = 16 + 4 + 4 + CDRHeaderSize;  Locator v1 = new Locator(IPAddress.Parse("10.20.30.40"), 2700);  SerializedPayload payload = new SerializedPayload();  payload.DataEncapsulation = EncapsulationManager.Serialize<Locator>(v1, Scheme);  IoBuffer buffer = IoBuffer.Wrap(payload.DataEncapsulation.SerializedPayload);  Assert.AreEqual(bufferSize, buffer.Remaining);  Assert.AreEqual("00 00 00 00 00 00 00 01 00 00 0A 8C 00 00 00 00 00 00 00 00 00 00 00 00 0A 14 1E 28", buffer.GetHexDump());  Locator v2 = EncapsulationManager.Deserialize<Locator>(buffer);  Assert.AreEqual(v1, v2);  } |
| ***Salida*** | Nombre de la prueba: TestExistConfiguration  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00,0286387 |

#### Prueba de Mensajes

### Transporte

#### Prueba de Detección de paquetes RTPS.

|  |  |
| --- | --- |
| **Llamada:**  public UDPReceiver(Uri uri, int bufferSize) | |
| ***Descripción*** | En esta prueba se verifica el correcto funcionamiento de los receptores UDP, utilizando mensajes RTPS, a los cuales se verifica que sus datos sean correctos con pequeñas pruebas assert |
| ***Entrada*** | Inicialmente no se tiene inicializado al *Receiver UDP* |
| ***Referencia*** |  |
| ***Código*** | [TestMethod]  public void TestPublishData()  {  object key = new object();  UDPReceiver rec = new UDPReceiver(new Uri("udp://" + Host + ":" + Port), 1024);  rec.MessageReceived += (s, m) =>  {  Message msg = m.Message;  Debug.WriteLine("New Message has arrived from {0}", m.Session.RemoteEndPoint);  Debug.WriteLine("Message Header: {0}", msg.Header);  Assert.AreEqual(ProtocolId.PROTOCOL\_RTPS, msg.Header.Protocol);  Assert.AreEqual(VendorId.OCI, msg.Header.VendorId);  Assert.AreEqual(ProtocolVersion.PROTOCOLVERSION\_2\_1, msg.Header.Version);  Assert.AreEqual(2, msg.SubMessages.Count);  foreach (var submsg in msg.SubMessages)  {  Debug.WriteLine("SubMessage: {0}", submsg);  if (submsg is Data)  {  Data d = submsg as Data;  foreach (var par in d.InlineQos.Value)  Debug.WriteLine("InlineQos: {0}", par);  }  }  lock (key) Monitor.Pulse(key);  };  rec.Start();  simulator.SendUDPPacket("SamplePackets/packet1.dat", Host, Port);  lock (key)  {  Assert.IsTrue(Monitor.Wait(key, 1000), "Time-out. Message has not arrived or there is an error on it.");  }  rec.Close();  }  } |
| ***Salida*** | Nombre de la prueba: *TestPublishData*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00,640815  Salida estándar de Result:  Trace du débogage :  no configuration section <common/logging> found - suppressing logging output  Sent 613/613 bytes to 224.0.1.111:9999  New Message has arrived from 172.30.82.26:56951  Message Header: [RTPS, 2.1, 01-03, 01-03-00-00-01-23-45-67-89-AB-CD-EF]  SubMessage: InfoTimestamp:header[9, 1, 8], 12/12/1944 16:04:37 [1418400277:3853715240]  SubMessage: Data:header[21, 7, 577], Payload[Rtps.Messages.Submessages.Elements.SerializedPayload]  InlineQos: ParameterId=PID\_KEY\_HASH, Content=09-23-09-23-00-00-00-00-00-00-00-00-00-00-00-00  InlineQos: ParameterId=PID\_SENTINEL, Content= |

#### Pruebas de paquetes RTPS.

|  |  |
| --- | --- |
| **Llamada:**  public UDPReceiver(Uri uri, int bufferSize) | |
| ***Descripción*** | En esta prueba se verifica el correcto funcionamiento de los receptores UDP, utilizando mensajes RTPS de otros vendors con los cuales se verifica que sus datos sean correctos con pequeñas pruebas assert |
| ***Entrada*** | Inicialmente no se tiene inicializado al *Receiver UDP* |
| ***Código*** | [TestMethod]  public void TestPublishPacket2()  {  object key = new object();  UDPReceiver rec = new UDPReceiver(new Uri("udp://" + Host + ":" + Port), 1024);  rec.MessageReceived += (s, m) =>  {  Message msg = m.Message;  Debug.WriteLine("New Message has arrived from {0}", m.Session.RemoteEndPoint);  Debug.WriteLine("Message Header: {0}", msg.Header);  Assert.AreEqual(ProtocolId.PROTOCOL\_RTPS, msg.Header.Protocol);  Assert.AreEqual(VendorId.OCI, msg.Header.VendorId);  Assert.AreEqual(ProtocolVersion.PROTOCOLVERSION\_2\_1, msg.Header.Version);  Assert.AreEqual(2, msg.SubMessages.Count);  foreach (var submsg in msg.SubMessages)  {  Debug.WriteLine("SubMessage: {0}", submsg);  switch (submsg.Kind)  {  case SubMessageKind.DATA:  Data d = submsg as Data;  foreach (var par in d.InlineQos.Value)  Debug.WriteLine("InlineQos: {0}", par);  break;  case SubMessageKind.INFO\_TS:  InfoTimestamp its = submsg as InfoTimestamp;  Debug.WriteLine("The TimeStampFlag value state is: {0}", its.HasInvalidateFlag);  Debug.WriteLine("The EndiannessFlag value state is: {0}", its.Header.Flags.IsLittleEndian);  Debug.WriteLine("The octetsToNextHeader value is: {0}", its.Header.SubMessageLength);  if (its.HasInvalidateFlag == false)  {  Debug.WriteLine("The Timestamp value is: {0}", its.TimeStamp);  }  break;  default:  Assert.Fail("Only Timestamp and Data submesages are expected");  break;  }  }  lock (key) Monitor.Pulse(key);  };  rec.Start();  simulator.SendUDPPacket("SamplePackets/packet3.dat", Host, Port);  lock (key)  {  Assert.IsTrue(Monitor.Wait(key, 10000), "Time-out. Message has not arrived or there is an error on it.");  }  rec.Close();  } |
| ***Salida*** | Nombre de la prueba: *TestPublishPacket2*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00,2431361  Salida estándar de Result:  Trace du débogage :  no configuration section <common/logging> found - suppressing logging output  Sent 196/196 bytes to 224.0.1.111:7400  New Message has arrived from 172.30.82.26:63179  Message Header: [RTPS, 2.1, 01-03, 01-03-00-00-01-23-45-67-89-AB-CD-EF]  SubMessage: InfoTimestamp:header[9, 1, 8], 01/01/1900 00:00:00 [0:0]  The TimeStampFlag value state is: False  The EndiannessFlag value state is: True  The octetsToNextHeader value is: 8  The Timestamp value is: 01/01/1900 00:00:00 [0:0]  SubMessage: Data:header[21, 11, 0], Payload[Rtps.Messages.Submessages.Elements.SerializedPayload]  InlineQos: ParameterId=PID\_STATUS\_INFO, Content=00-00-00-01  InlineQos: ParameterId=PID\_TOPIC\_NAME, Content=0A-00-00-00-4D-79-20-54-6F-70-69-63-20-00-00-00  InlineQos: ParameterId=PID\_PRESENTATION, Content=E7-03-00-00-00-00-00-00  InlineQos: ParameterId=PID\_PARTITION, Content=01-00-00-00-06-00-00-00-48-65-6C-6C-6F-00-00-00  InlineQos: ParameterId=PID\_OWNERSHIP\_STRENGTH, Content=0C-00-00-00  InlineQos: ParameterId=PID\_LIVELINESS, Content=02-00-00-00-FF-FF-FF-7F-FF-FF-FF-7F  InlineQos: ParameterId=PID\_RELIABILITY, Content=00-00-00-00-00-00-00-00-00-E1-F5-05  InlineQos: ParameterId=PID\_TRANSPORT\_PRIORITY, Content=0D-00-00-00  InlineQos: ParameterId=PID\_LIFESPAN, Content=0E-00-00-00-FF-FF-FF-7F  InlineQos: ParameterId=PID\_DESTINATION\_ORDER, Content=01-00-00-00  InlineQos: ParameterId=PID\_SENTINEL, Content= |

|  |  |
| --- | --- |
| **Llamada:**  public UDPReceiver(Uri uri, int bufferSize) | |
| ***Descripción*** | En esta prueba se verifica el correcto funcionamiento de los receptores UDP, utilizando mensajes RTPS de otros vendors con los cuales se verifica que sus datos sean correctos con pequeñas pruebas assert. De esta prueba se pueden derivar otras. |
| ***Entrada*** | Inicialmente no se tiene inicializado al *Receiver UDP* |
| ***Referencia*** |  |
| ***Código*** | [TestMethod]  public void GeneralRTPSMessageTesterMethod()  {  object key = new object();  UDPReceiver rec = new UDPReceiver(new Uri("udp://" + Host + ":" + Port), 1024);  rec.MessageReceived += (s, m) =>  {  Message msg = m.Message;  Debug.WriteLine("New Message has arrived from {0}", m.Session.RemoteEndPoint);  Debug.WriteLine("Message Header: {0}", msg.Header);  Assert.AreEqual(ProtocolId.PROTOCOL\_RTPS, msg.Header.Protocol);  Debug.WriteLine("The Header Protocol is: {0}", msg.Header.Protocol);  Assert.AreEqual(VendorId.OCI, msg.Header.VendorId);  Debug.WriteLine("The VendorId value state is: {0}", msg.Header.VendorId);  Assert.AreEqual(ProtocolVersion.PROTOCOLVERSION\_2\_1, msg.Header.Version);  Debug.WriteLine("The Protocol Version value state is: {0}", msg.Header.Version);  Debug.WriteLine("The number of SubMessages in the message is: {0}", msg.SubMessages.Count);  //Assert.AreEqual(2, msg.SubMessages.Count);  foreach (var submsg in msg.SubMessages)  {  Debug.WriteLine("SubMessage: {0}", submsg.Kind);  switch (submsg.Kind)  {  case SubMessageKind.DATA:  {  Data d = submsg as Data;    Debug.WriteLine("The KeyFlag value state is: {0}", d.HasKeyFlag);  Debug.WriteLine("The DataFlag value state is: {0}", d.HasDataFlag);  Debug.WriteLine("The InlineQoSFlag value state is: {0}", d.HasInlineQosFlag);  Debug.WriteLine("The EndiannessFlag value state is: {0}", d.Header.Flags.IsLittleEndian);  Debug.WriteLine("The octetsToNextHeader value is: {0}", d.Header.SubMessageLength);  Debug.WriteLine("The extraFlags value is: {0}", d.ExtraFlags.Value);  Debug.WriteLine("The octetsToInlineQos value is: Aun no logro");  Debug.WriteLine("The readerID is: {0}", d.ReaderId);  Debug.WriteLine("The writerID is: {0}", d.WriterId);  Debug.WriteLine("The writerSN is: {0}", d.WriterSN);  if (d.HasInlineQosFlag)  {  foreach (var par in d.InlineQos.Value)  {  Debug.WriteLine("InlineQos: {0}", par);  }  }  if (d.HasDataFlag || d.Header.Flags.IsLittleEndian)  {  for (int i = 0; i <= d.SerializedPayload.DataEncapsulation.SerializedPayload.Length - 1; i++)  {  Debug.WriteLine("SerializedPayload: {0}", d.SerializedPayload.DataEncapsulation.SerializedPayload.GetValue(i));  }  }  break;  }  case SubMessageKind.ACKNACK:  {  AckNack d = submsg as AckNack;  Debug.WriteLine("The FinalFlag value state is: {0}", d.HasFinalFlag);  Debug.WriteLine("The EndiannessFlag value state is: {0}", d.Header.Flags.IsLittleEndian);  Debug.WriteLine("The octetsToNextHeader value is: {0}", d.Header.SubMessageLength);  Debug.WriteLine("The readerID is: {0}", d.ReaderId);  Debug.WriteLine("The writerID is: {0}", d.WriterId);  Debug.WriteLine("The readerSNState is: {0}", d.ReaderSNState);  Debug.WriteLine("The Count is: {0}", d.Count);  break;  }  case SubMessageKind.NACK\_FRAG:  {  NackFrag d = submsg as NackFrag;  Debug.WriteLine("The EndiannessFlag value state is: {0}", d.Header.Flags.IsLittleEndian);  Debug.WriteLine("The octetsToNextHeader value is: {0}", d.Header.SubMessageLength);  Debug.WriteLine("The readerID is: {0}", d.ReaderId);  Debug.WriteLine("The writerID is: {0}", d.WriterId);  Debug.WriteLine("The writerSN is: {0}", d.WriterSequenceNumber);  Debug.WriteLine("The fragmentNumberState value is: {0}", d.FragmentNumberState);  break;  }  case SubMessageKind.DATA\_FRAG:  {  DataFrag d = submsg as DataFrag;  Debug.WriteLine("The KeyFlag value state is: {0}", d.HasKeyFlag);  Debug.WriteLine("The InlineQoSFlag value state is: {0}", d.HasInlineQosFlag);  Debug.WriteLine("The EndiannessFlag value state is: {0}", d.Header.Flags.IsLittleEndian);  Debug.WriteLine("The octetsToNextHeader value is: {0}", d.Header.SubMessageLength);  Debug.WriteLine("The extraFlags value is: {0}", d.ExtraFlags);  Debug.WriteLine("The octetsToInlineQos value is: Aun no logro");  Debug.WriteLine("The readerID is: {0}", d.ReaderId);  Debug.WriteLine("The writerID is: {0}", d.WriterId);  Debug.WriteLine("The writerSN is: {0}", d.WriterSequenceNumber);  Debug.WriteLine("The FragmentNumber is: {0}", d.FragmentStartingNumber);  Debug.WriteLine("The fragmentsInSubmessage is: {0}", d.FragmentsInSubmessage);  Debug.WriteLine("The samplesize is: {0}", d.SampleSize);  if (d.HasInlineQosFlag)  {  foreach (var par in d.ParameterList.Value)  {  Debug.WriteLine("InlineQos: {0}", par);  }  }  for (int i = 0; i <= d.SerializedPayload.Length - 1; i++)  {  Debug.WriteLine("SerializedPayload: {0}", d.SerializedPayload.GetValue(i));  }  break;  }  case SubMessageKind.GAP:  {  Gap d = submsg as Gap;  Debug.WriteLine("The EndiannessFlag value state is: {0}", d.Header.Flags.IsLittleEndian);  Debug.WriteLine("The octetsToNextHeader value is: {0}", d.Header.SubMessageLength);  Debug.WriteLine("The readerID is: {0}", d.ReaderId);  Debug.WriteLine("The writerID is: {0}", d.WriterId);  Debug.WriteLine("The GapStart number is: {0}", d.GapStart);  Debug.WriteLine("The GapList value is: {0}", d.GapList);  break;  }  case SubMessageKind.HEARTBEAT:  {  Heartbeat d = submsg as Heartbeat;  Debug.WriteLine("The LivelinessFlag value state is: {0}", d.HasLivelinessFlag);  Debug.WriteLine("The FinalFlag value state is: {0}", d.HasFinalFlag);  Debug.WriteLine("The EndiannessFlag value state is: {0}", d.Header.Flags.IsLittleEndian);  Debug.WriteLine("The octetsToNextHeader value is: {0}", d.Header.SubMessageLength);  Debug.WriteLine("The readerID is: {0}", d.ReaderId);  Debug.WriteLine("The writerID is: {0}", d.WriterId);  Debug.WriteLine("The firstSN is: {0}", d.FirstSequenceNumber);  Debug.WriteLine("The lastSN is: {0}", d.LastSequenceNumber);  Debug.WriteLine("The Count is: {0}", d.Count);  break;  }  case SubMessageKind.HEARTBEAT\_FRAG:  {  HeartbeatFrag d = submsg as HeartbeatFrag;  Debug.WriteLine("The EndiannessFlag value state is: {0}", d.Header.Flags.IsLittleEndian);  Debug.WriteLine("The octetsToNextHeader value is: {0}", d.Header.SubMessageLength);  Debug.WriteLine("The readerID is: {0}", d.ReaderId);  Debug.WriteLine("The writerID is: {0}", d.WriterId);  Debug.WriteLine("The writerSN is: {0}", d.WriterSequenceNumber);  Debug.WriteLine("The FragmentNumber is: {0}", d.LastFragmentNumber);  Debug.WriteLine("The Count is: {0}", d.Count);  break;  }  case SubMessageKind.INFO\_DST:  {  InfoDestination d = submsg as InfoDestination;  Debug.WriteLine("The EndiannessFlag value state is: {0}", d.Header.Flags.IsLittleEndian);  Debug.WriteLine("The octetsToNextHeader value is: {0}", d.Header.SubMessageLength);  Debug.WriteLine("The guidPrefix value is: {0}", d.GuidPrefix);  break;  }  case SubMessageKind.INFO\_TS:  {  InfoTimestamp d = submsg as InfoTimestamp;  Debug.WriteLine("The TimeStampFlag value state is: {0}", d.HasInvalidateFlag);  Debug.WriteLine("The EndiannessFlag value state is: {0}", d.Header.Flags.IsLittleEndian);  Debug.WriteLine("The octetsToNextHeader value is: {0}", d.Header.SubMessageLength);  if (d.HasInvalidateFlag == false)  {  Debug.WriteLine("The Timestamp value is: {0}", d.TimeStamp);  }  break;  }  case SubMessageKind.INFO\_SRC:  {  InfoSource d = submsg as InfoSource;  Debug.WriteLine("The EndiannessFlag value state is: {0}", d.Header.Flags.IsLittleEndian);  Debug.WriteLine("The octetsToNextHeader value is: {0}", d.Header.SubMessageLength);  Debug.WriteLine("The ProtocolVersion value is: {0}", d.ProtocolVersion);  Debug.WriteLine("The vendorId value is: {0}", d.VendorId);  Debug.WriteLine("The guidPrefix value is: {0}", d.GuidPrefix);  break;  }  case SubMessageKind.INFO\_REPLY:  {  InfoReply d = submsg as InfoReply;  Debug.WriteLine("The EndiannessFlag value state is: {0}", d.Header.Flags.IsLittleEndian);  Debug.WriteLine("The octetsToNextHeader value is: {0}", d.Header.SubMessageLength);  Debug.WriteLine("The MulticastFlag value state is: {0}", d.HasMulticastFlag);  Debug.WriteLine("The unicastLocatorList value state is: {0}", d.UnicastLocatorList);  if (d.HasMulticastFlag)  {  Debug.WriteLine("The multicastLocatorList value state is: {0}", d.MulticastLocatorList);  }  break;  }  case SubMessageKind.INFO\_REPLY\_IP4:  {  InfoReplyIp4 d = submsg as InfoReplyIp4;  Debug.WriteLine("The EndiannessFlag value state is: {0}", d.Header.Flags.IsLittleEndian);  Debug.WriteLine("The octetsToNextHeader value is: {0}", d.Header.SubMessageLength);  Debug.WriteLine("The MulticastFlag value state is: {0}", d.HasMulticastFlag);  Debug.WriteLine("The unicastLocatorList value state is: {0}", d.UnicastLocator);  if (d.HasMulticastFlag)  {  Debug.WriteLine("The multicastLocatorList value state is: {0}", d.MulticastLocator);  }  break;  }  case SubMessageKind.PAD:  {  Pad d = submsg as Pad;  Debug.WriteLine("The EndiannessFlag value state is: {0}", d.Header.Flags.IsLittleEndian);  Debug.WriteLine("The octetsToNextHeader value is: {0}", d.Header.SubMessageLength);  break;  }  }  }  lock (key) Monitor.Pulse(key);  };  rec.Start();  simulator.SendUDPPacket("SamplePackets/TestOpenDDS\_rtps\_reliability\_runtest\_local/Packet04.dat", Host, Port);  lock (key)  {  Assert.IsTrue(Monitor.Wait(key, 10000), "Time-out. Message has not arrived or there is an error on it.");  }  rec.Close();  } |
| ***Salida*** | Nombre de la prueba: *GeneralRTPSMessageTesterMethod*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00,2446508  Nombre de la prueba: GeneralRTPSMessageTesterMethod  Resultado de la prueba: Superada  Salida estándar de Result:  Trace du débogage :  no configuration section <common/logging> found - suppressing logging output  Sent 52/52 bytes to 224.0.1.111:7400  New Message has arrived from 172.30.82.26:54070  Message Header: [RTPS, 2.1, 01-03, 01-03-08-00-27-B9-29-47-0A-AF-00-00]  The Header Protocol is: RTPS  The VendorId value state is: 01-03  The Protocol Version value state is: 2.1  The number of SubMessages in the message is: 1  SubMessage: DATA  The KeyFlag value state is: False  The DataFlag value state is: True  The InlineQoSFlag value state is: False  The EndiannessFlag value state is: True  The octetsToNextHeader value is: 0  The extraFlags value is: 0  The octetsToInlineQos value is: Aun no logro  The readerID is: 0-USER\_DEFINED\_UNKNOWN  The writerID is: 3-USER\_DEFINED\_WRITER\_W\_KEY  The writerSN is: 3  SerializedPayload: 205  SerializedPayload: 171  SerializedPayload: 205  SerializedPayload: 171 |

|  |  |
| --- | --- |
| **Llamada:**  public UDPReceiver(Uri uri, int bufferSize) | |
| ***Descripción*** | En esta prueba se verifica el correcto funcionamiento de los receptores UDP, utilizando mensajes RTPS de otros vendors con los cuales se verifica que sus datos sean correctos con pequeñas pruebas assert. |
| ***Entrada*** | Inicialmente no se tiene inicializado al *Receiver UDP* |
| ***Referencia*** |  |
| ***Código*** | [TestMethod]  public void TesOpenDDS\_rtps\_reliability\_runtest\_localPacket01()  {  object key = new object();  UDPReceiver rec = new UDPReceiver(new Uri("udp://" + Host + ":" + Port), 1024);  rec.MessageReceived += (s, m) =>  {  Message msg = m.Message;  Debug.WriteLine("New Message has arrived from {0}", m.Session.RemoteEndPoint);    Debug.WriteLine("Message Header: {0}", msg.Header);  Assert.AreEqual(ProtocolId.PROTOCOL\_RTPS.ToString(), msg.Header.Protocol.ToString());  Debug.WriteLine("The Header Protocol is: {0}", msg.Header.Protocol);  Assert.AreEqual(ProtocolVersion.PROTOCOLVERSION\_2\_1.ToString(), msg.Header.Version.ToString());  Debug.WriteLine("The Protocol Version value state is: {0}", msg.Header.Version);  Assert.AreEqual(VendorId.OCI.ToString(), msg.Header.VendorId.ToString());  Debug.WriteLine("The VendorId value state is: {0}", msg.Header.VendorId);  Assert.AreEqual("01-03-08-00-27-B9-29-47-0A-AF-00-00", msg.Header.GuidPrefix.ToString());  Debug.WriteLine("The guidPrefix value state is: {0}", msg.Header.GuidPrefix);  Assert.AreEqual(1, msg.SubMessages.Count);  Debug.WriteLine("The number of SubMessages in the message is: {0}", msg.SubMessages.Count);    foreach (var submsg in msg.SubMessages)  {  Assert.AreEqual(SubMessageKind.DATA, submsg.Kind );  Debug.WriteLine("SubMessage: {0}", submsg.Kind);    switch (submsg.Kind)  {  case SubMessageKind.DATA:  {  Data d = submsg as Data;    Assert.AreEqual(false, d.HasKeyFlag);  Debug.WriteLine("The KeyFlag value state is: {0}", d.HasKeyFlag);  Assert.AreEqual(true, d.HasDataFlag);  Debug.WriteLine("The DataFlag value state is: {0}", d.HasDataFlag);  Assert.AreEqual(false, d.HasInlineQosFlag);  Debug.WriteLine("The InlineQoSFlag value state is: {0}", d.HasInlineQosFlag);  Assert.AreEqual(true, d.Header.Flags.IsLittleEndian);  Debug.WriteLine("The EndiannessFlag value state is: {0}", d.Header.Flags.IsLittleEndian);  Assert.AreEqual(0, d.Header.SubMessageLength);  Debug.WriteLine("The octetsToNextHeader value is: {0}", d.Header.SubMessageLength);  Assert.AreEqual(0, d.ExtraFlags .Value);  Debug.WriteLine("The extraFlags value is: {0}", d.ExtraFlags.Value);  Debug.WriteLine("The octetsToInlineQos value is: ");  Assert.AreEqual(0, d.ReaderId.EntityKey0);  Assert.AreEqual(0, d.ReaderId.EntityKey1);  Assert.AreEqual(0, d.ReaderId.EntityKey2);  Debug.WriteLine("The readerIDEntityKey is: {0}-{1}-{2}", d.ReaderId.EntityKey0,d.ReaderId.EntityKey1,d.ReaderId.EntityKey2);  Assert.AreEqual(0,(int) d.ReaderId.TypeID);  Debug.WriteLine("The readerIDEntityKind value is: {0} ",(int)d.ReaderId.TypeID);  Assert.AreEqual(0, d.WriterId.EntityKey0);  Assert.AreEqual(1, d.WriterId.EntityKey1);  Assert.AreEqual(2, d.WriterId.EntityKey2);  Debug.WriteLine("The writerID is: {0}-{1}-{2}", d.WriterId.EntityKey0, d.WriterId.EntityKey1, d.WriterId.EntityKey2);  Assert.AreEqual(2, (int)d.WriterId.TypeID);  Debug.WriteLine("The writerIDEntityKind value is:{0} ",(int) d.WriterId.TypeID);  Assert.AreEqual("1", d.WriterSN.ToString());    Debug.WriteLine("The writerSN is: {0}", d.WriterSN);  if (d.HasInlineQosFlag)  {  /\*foreach (var par in d.InlineQos.Value)  {  Debug.WriteLine("InlineQos: {0}", par);  }\*/  }      if (d.HasDataFlag || d.Header.Flags.IsLittleEndian)  {  for (int i = 0; i <= d.SerializedPayload.DataEncapsulation.SerializedPayload.Length - 1; i++)  {    Debug.WriteLine("SerializedPayload: {0}", d.SerializedPayload.DataEncapsulation.SerializedPayload.GetValue(i));  }  }  break;  }  }  }  lock (key) Monitor.Pulse(key);  };  rec.Start();  simulator.SendUDPPacket("SamplePackets/TestOpenDDS\_rtps\_reliability\_runtest\_local/Packet01.dat", Host, Port);  lock (key)  {  Assert.IsTrue(Monitor.Wait(key, 10000), "Time-out. Message has not arrived or there is an error on it.");  }  rec.Close();  } |
| ***Salida*** | Nombre de la prueba: *TestOpenDDS\_rtps\_reliability\_runtest\_localPacket01*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00,2312824  Salida estándar de Result:  Trace du débogage :  no configuration section <common/logging> found - suppressing logging output  Sent 52/52 bytes to 224.0.1.111:7400  New Message has arrived from 172.30.82.26:63701  Message Header: [RTPS, 2.1, 01-03, 01-03-08-00-27-B9-29-47-0A-AF-00-00]  The Header Protocol is: RTPS  The Protocol Version value state is: 2.1  The VendorId value state is: 01-03  The guidPrefix value state is: 01-03-08-00-27-B9-29-47-0A-AF-00-00  The number of SubMessages in the message is: 1  SubMessage: DATA  The KeyFlag value state is: False  The DataFlag value state is: True  The InlineQoSFlag value state is: False  The EndiannessFlag value state is: True  The octetsToNextHeader value is: 0  The extraFlags value is: 0  The octetsToInlineQos value is:  The readerIDEntityKey is: 0-0-0  The readerIDEntityKind value is: 0  The writerID is: 0-1-2  The writerIDEntityKind value is:2  The writerSN is: 1  SerializedPayload: 205  SerializedPayload: 171  SerializedPayload: 205  SerializedPayload: 171 |

|  |  |
| --- | --- |
| **Llamada:**  public UDPReceiver(Uri uri, int bufferSize) | |
| ***Descripción*** | En esta prueba se verifica el correcto funcionamiento de los receptores UDP, utilizando mensajes RTPS de otros vendors con los cuales se verifica que sus datos sean correctos con pequeñas pruebas assert. |
| ***Entrada*** | Inicialmente no se tiene inicializado al *Receiver UDP* |
| ***Referencia*** |  |
| ***Código*** | [TestMethod]  public void TesOpenDDS\_rtps\_reliability\_runtest\_localPacket02()  {  object key = new object();  UDPReceiver rec = new UDPReceiver(new Uri("udp://" + Host + ":" + Port), 1024);  rec.MessageReceived += (s, m) =>  {  Message msg = m.Message;  Debug.WriteLine("New Message has arrived from {0}", m.Session.RemoteEndPoint);  Debug.WriteLine("Message Header: {0}", msg.Header);  Assert.AreEqual(ProtocolId.PROTOCOL\_RTPS.ToString(), msg.Header.Protocol.ToString());  Debug.WriteLine("The Header Protocol is: {0}", msg.Header.Protocol);  Assert.AreEqual(ProtocolVersion.PROTOCOLVERSION\_2\_1.ToString(), msg.Header.Version.ToString());  Debug.WriteLine("The Protocol Version value state is: {0}", msg.Header.Version);  Assert.AreEqual(VendorId.OCI.ToString(), msg.Header.VendorId.ToString());  Debug.WriteLine("The VendorId value state is: {0}", msg.Header.VendorId);  Assert.AreEqual("01-03-08-00-27-B9-29-47-0A-AF-00-00", msg.Header.GuidPrefix.ToString());  Debug.WriteLine("The guidPrefix value state is: {0}", msg.Header.GuidPrefix);  Assert.AreEqual(1, msg.SubMessages.Count);  Debug.WriteLine("The number of SubMessages in the message is: {0}", msg.SubMessages.Count);  foreach (var submsg in msg.SubMessages)  {  Assert.AreEqual(SubMessageKind.HEARTBEAT, submsg.Kind);  Debug.WriteLine("SubMessage: {0}", submsg.Kind);  switch (submsg.Kind)  {  case SubMessageKind.HEARTBEAT:  {  Heartbeat d = submsg as Heartbeat;  Assert.AreEqual(false, d.HasLivelinessFlag);  Debug.WriteLine("The LivelinessFlag value state is: {0}", d.HasLivelinessFlag);  Assert.AreEqual(false, d.HasFinalFlag);  Debug.WriteLine("The FinalFlag value state is: {0}", d.HasFinalFlag);  Assert.AreEqual(true, d.Header.Flags.IsLittleEndian);  Debug.WriteLine("The EndiannessFlag value state is: {0}", d.Header.Flags.IsLittleEndian);  Assert.AreEqual(0, d.Header.SubMessageLength);  Debug.WriteLine("The octetsToNextHeader value is: {0}", d.Header.SubMessageLength);  Assert.AreEqual(0, d.ReaderId.EntityKey0);  Assert.AreEqual(0, d.ReaderId.EntityKey1);  Assert.AreEqual(0, d.ReaderId.EntityKey2);  Debug.WriteLine("The readerIDEntityKey is: {0}-{1}-{2}", d.ReaderId.EntityKey0, d.ReaderId.EntityKey1, d.ReaderId.EntityKey2);  Assert.AreEqual(0, (int)d.ReaderId.TypeID);  Debug.WriteLine("The readerIDEntityKind value is: {0} ", (int)d.ReaderId.TypeID);  Assert.AreEqual(0, d.WriterId.EntityKey0);  Assert.AreEqual(1, d.WriterId.EntityKey1);  Assert.AreEqual(2, d.WriterId.EntityKey2);  Debug.WriteLine("The writerID is: {0}-{1}-{2}", d.WriterId.EntityKey0, d.WriterId.EntityKey1, d.WriterId.EntityKey2);  Assert.AreEqual(2, (int)d.WriterId.TypeID);    Debug.WriteLine("The writerIDEntityKind value is:{0} ", (int)d.WriterId.TypeID);  Assert.AreEqual(1,d.FirstSequenceNumber);  Debug.WriteLine("The firstSN is: {0}", d.FirstSequenceNumber);  Assert.AreEqual(1,d.LastSequenceNumber);  Debug.WriteLine("The lastSN is: {0}", d.LastSequenceNumber);  Assert.AreEqual(1,d.Count);  Debug.WriteLine("The Count is: {0}", d.Count);  break;  }    }  }  lock (key) Monitor.Pulse(key);  };  rec.Start();  simulator.SendUDPPacket("SamplePackets/TestOpenDDS\_rtps\_reliability\_runtest\_local/Packet02.dat", Host, Port);  lock (key)  {  Assert.IsTrue(Monitor.Wait(key, 10000), "Time-out. Message has not arrived or there is an error on it.");  }  rec.Close();  } |
| ***Salida*** | Nombre de la prueba: *TestOpenDDS\_rtps\_reliability\_runtest\_localPacket02*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00,2411492  Nombre de la prueba: TesOpenDDS\_rtps\_reliability\_runtest\_localPacket02  Resultado de la prueba: Superada  Salida estándar de Result:  Trace du débogage :  no configuration section <common/logging> found - suppressing logging output  Sent 52/52 bytes to 224.0.1.111:7400  New Message has arrived from 172.30.82.26:59411  Message Header: [RTPS, 2.1, 01-03, 01-03-08-00-27-B9-29-47-0A-AF-00-00]  The Header Protocol is: RTPS  The Protocol Version value state is: 2.1  The VendorId value state is: 01-03  The guidPrefix value state is: 01-03-08-00-27-B9-29-47-0A-AF-00-00  The number of SubMessages in the message is: 1  SubMessage: HEARTBEAT  The LivelinessFlag value state is: False  The FinalFlag value state is: False  The EndiannessFlag value state is: True  The octetsToNextHeader value is: 0  The readerIDEntityKey is: 0-0-0  The readerIDEntityKind value is: 0  The writerID is: 0-1-2  The writerIDEntityKind value is:2  The firstSN is: 1  The lastSN is: 1  The Count is: 1 |

|  |  |
| --- | --- |
| **Llamada:**  public UDPReceiver(Uri uri, int bufferSize) | |
| ***Descripción*** | En esta prueba se verifica el correcto funcionamiento de los receptores UDP, utilizando mensajes RTPS de otros vendors con los cuales se verifica que sus datos sean correctos con pequeñas pruebas assert. |
| ***Entrada*** | Inicialmente no se tiene inicializado al *Receiver UDP* |
| ***Referencia*** |  |
| ***Código*** | [TestMethod]  public void TesOpenDDS\_rtps\_reliability\_runtest\_localPacket03()  {  object key = new object();  UDPReceiver rec = new UDPReceiver(new Uri("udp://" + Host + ":" + Port), 1024);  rec.MessageReceived += (s, m) =>  {  Message msg = m.Message;  Debug.WriteLine("New Message has arrived from {0}", m.Session.RemoteEndPoint);  Debug.WriteLine("Message Header: {0}", msg.Header);  Assert.AreEqual(ProtocolId.PROTOCOL\_RTPS.ToString(), msg.Header.Protocol.ToString());  Debug.WriteLine("The Header Protocol is: {0}", msg.Header.Protocol);  Assert.AreEqual(ProtocolVersion.PROTOCOLVERSION\_2\_1.ToString(), msg.Header.Version.ToString());  Debug.WriteLine("The Protocol Version value state is: {0}", msg.Header.Version);  Assert.AreEqual(VendorId.OCI.ToString(), msg.Header.VendorId.ToString());  Debug.WriteLine("The VendorId value state is: {0}", msg.Header.VendorId);  Assert.AreEqual("01-03-08-00-27-B9-29-47-0A-AF-00-01", msg.Header.GuidPrefix.ToString());  Debug.WriteLine("The guidPrefix value state is: {0}", msg.Header.GuidPrefix);  Assert.AreEqual(2, msg.SubMessages.Count);  Debug.WriteLine("The number of SubMessages in the message is: {0}", msg.SubMessages.Count);  foreach (var submsg in msg.SubMessages)  {    Debug.WriteLine("SubMessage: {0}", submsg.Kind);  switch (submsg.Kind)  {  case SubMessageKind.INFO\_DST:  {  InfoDestination d = submsg as InfoDestination;  Assert.AreEqual(true, d.Header.Flags.IsLittleEndian);  Debug.WriteLine("The EndiannessFlag value state is: {0}", d.Header.Flags.IsLittleEndian);  Assert.AreEqual(12, d.Header.SubMessageLength);  Debug.WriteLine("The octetsToNextHeader value is: {0}", d.Header.SubMessageLength);  Assert.AreEqual("01-03-08-00-27-B9-29-47-0A-AF-00-00", d.GuidPrefix.ToString());  Debug.WriteLine("The guidPrefix value is: {0}", d.GuidPrefix);  break;  }  case SubMessageKind.ACKNACK:  {  AckNack d = submsg as AckNack;  Assert.AreEqual(true, d.HasFinalFlag);  Debug.WriteLine("The FinalFlag value state is: {0}", d.HasFinalFlag);  Assert.AreEqual(true, d.Header.Flags.IsLittleEndian);  Debug.WriteLine("The EndiannessFlag value state is: {0}", d.Header.Flags.IsLittleEndian);  Debug.WriteLine("The octetsToNextHeader value is: {0}", d.Header.SubMessageLength);  Assert.AreEqual(0, d.ReaderId.EntityKey0);  Assert.AreEqual(1, d.ReaderId.EntityKey1);  Assert.AreEqual(5, d.ReaderId.EntityKey2);  Debug.WriteLine("The readerIDEntityKey is: {0}-{1}-{2}", d.ReaderId.EntityKey0, d.ReaderId.EntityKey1, d.ReaderId.EntityKey2);  Assert.AreEqual(7, (int)d.ReaderId.TypeID);  Debug.WriteLine("The readerIDEntityKind value is: {0} ", (int)d.ReaderId.TypeID);  Assert.AreEqual(0, d.WriterId.EntityKey0);  Assert.AreEqual(1, d.WriterId.EntityKey1);  Assert.AreEqual(2, d.WriterId.EntityKey2);  Debug.WriteLine("The writerID is: {0}-{1}-{2}", d.WriterId.EntityKey0, d.WriterId.EntityKey1, d.WriterId.EntityKey2);  Assert.AreEqual(2, (int)d.WriterId.TypeID);  Debug.WriteLine("The writerIDEntityKind value is:{0} ", (int)d.WriterId.TypeID);  Assert.AreEqual("2", d.ReaderSNState.BitmapBase.ToString());  Assert.AreEqual(1, d.ReaderSNState.NumBits);  Assert.AreEqual(0, d.ReaderSNState.Bitmaps[0]);    Debug.WriteLine("The readerSNState is: {0}", d.ReaderSNState);  Debug.WriteLine("The Count is: {0}", d.Count);  break;  }  }  }  lock (key) Monitor.Pulse(key);  };  rec.Start();  simulator.SendUDPPacket("SamplePackets/TestOpenDDS\_rtps\_reliability\_runtest\_local/Packet03.dat", Host, Port);  lock (key)  {  Assert.IsTrue(Monitor.Wait(key, 1000), "Time-out. Message has not arrived or there is an error on it.");  }  rec.Close();  }    } |
| ***Salida*** | Nombre de la prueba: *TestOpenDDS\_rtps\_reliability\_runtest\_localPacket03*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00,2464427  Salida estándar de Result:  Trace du débogage :  no configuration section <common/logging> found - suppressing logging output  Sent 68/68 bytes to 224.0.1.111:7400  New Message has arrived from 172.30.82.26:63366  Message Header: [RTPS, 2.1, 01-03, 01-03-08-00-27-B9-29-47-0A-AF-00-01]  The Header Protocol is: RTPS  The Protocol Version value state is: 2.1  The VendorId value state is: 01-03  The guidPrefix value state is: 01-03-08-00-27-B9-29-47-0A-AF-00-01  The number of SubMessages in the message is: 2  SubMessage: INFO\_DST  The EndiannessFlag value state is: True  The octetsToNextHeader value is: 12  The guidPrefix value is: 01-03-08-00-27-B9-29-47-0A-AF-00-00  SubMessage: ACKNACK  The FinalFlag value state is: True  The EndiannessFlag value state is: True  The octetsToNextHeader value is: 28  The readerIDEntityKey is: 0-1-5  The readerIDEntityKind value is: 7  The writerID is: 0-1-2  The writerIDEntityKind value is:2  The readerSNState is: 2/1:[0x0000]  The Count is: 1 |

### Utils

#### Pruebas del generador de identidad.

|  |  |
| --- | --- |
| **Llamada:**  static GuidGenerator() | |
| ***Descripción*** | En esta prueba se verifica el correcto funcionamiento del Guid Generator. |
| ***Entrada*** | Inicialmente no se tiene inicializado al *GuidGenerator* |
| ***Referencia*** |  |
| ***Código*** | [TestMethod]  public void TestGeneration1()  {  GuidGenerator generator = new GuidGenerator();  GUID guid = generator.GenerateGuid();  Assert.AreEqual(GuidGenerator.VENDORID\_DOOPEC[0], guid.Prefix.Prefix[0]);  Assert.AreEqual(GuidGenerator.VENDORID\_DOOPEC[1], guid.Prefix.Prefix[1]);  } |
| ***Salida*** | Nombre de la prueba: *TestGeneration1*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00,172692 |

|  |  |
| --- | --- |
| **Llamada:**  static GuidGenerator() | |
| ***Descripción*** | En esta prueba se verifica el correcto funcionamiento del Guid Generator. |
| ***Entrada*** | Inicialmente no se tiene inicializado al *GuidGenerator* |
| ***Referencia*** |  |
| ***Código*** | [TestMethod]  public void TestGeneration2()  {  GuidGenerator generator = new GuidGenerator();  GUID guid1 = generator.GenerateGuid();  GUID guid2 = generator.GenerateGuid();  Assert.AreNotEqual(guid2.Prefix.ToString(), guid1.Prefix.ToString());  } |
| ***Salida*** | Nombre de la prueba: *TestGeneration2*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00,1552052 |

#### Pruebas del PeriodicWorker.

|  |  |
| --- | --- |
| **Llamada:**  private void KeepWorkerRunning() | |
| ***Descripción*** | En esta prueba se verifica el correcto funcionamiento del Worker en el cual se realiza tareas de actualización y descubrimiento |
| ***Entrada*** | Inicialmente no se tiene inicializado al Worker |
| ***Referencia*** |  |
| ***Código*** | [TestMethod]  public void TestWorkerVerySlow()  {  int period = 2 \* 1000;  int sleepTime = 20 \* 1000+90;  PeriodicWorker worker = new PeriodicWorker();  worker.Start(period);  Thread.Sleep(sleepTime);  worker.End();  Assert.AreEqual(sleepTime / period, worker.Count);  } |
| ***Salida*** | Nombre de la prueba: *TestWorkerVerySlow*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:20,2217009 |

|  |  |
| --- | --- |
| **Llamada:**  private void KeepWorkerRunning() | |
| ***Descripción*** | En esta prueba se verifica el correcto funcionamiento del Worker en el cual se realiza tareas de actualización y descubrimiento |
| ***Entrada*** | Inicialmente no se tiene inicializado al Worker |
| ***Referencia*** |  |
| ***Código*** | [TestMethod]  public void TestWorkerSlow()  {  int period = 2 \* 100;  int sleepTime = 20 \* 100 + 50;  PeriodicWorker worker = new PeriodicWorker();  worker.Start(period);  Thread.Sleep(sleepTime);  worker.End();  Assert.AreEqual(sleepTime / period, worker.Count);  } |
| ***Salida*** | Nombre de la prueba: *TestWorkerSlow*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:02,1763206 |

|  |  |
| --- | --- |
| **Llamada:**  private void KeepWorkerRunning() | |
| ***Descripción*** | En esta prueba se verifica el correcto funcionamiento del Worker en el cual se realiza tareas de actualización y descubrimiento |
| ***Entrada*** | Inicialmente no se tiene inicializado al Worker |
| ***Referencia*** |  |
| ***Código*** | [TestMethod]  public void TestWorkerQuick()  {  int period = 2 \* 10;  int sleepTime = 20 \* 10 + 50;  PeriodicWorker worker = new PeriodicWorker();  worker.Start(period);  Thread.Sleep(sleepTime);  worker.End();  Assert.AreEqual(sleepTime / period, worker.Count);  } |
| ***Salida*** | Nombre de la prueba: *TestWorkerQuick*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00,3822312 |

#### Prueba de tiempo

|  |  |
| --- | --- |
| **Llamada:**  public Time(long systemCurrentMillis) | |
| ***Descripción*** | En esta prueba se verifica el correcto funcionamiento del temporizador |
| ***Entrada*** | Inicialmente no se tiene inicializado al Time |
| ***Referencia*** |  |
| ***Código*** | [TestMethod]  public void TestTimeSeconds()  {  long timeMillis = 1000; // 1 sec  Time t = new Time(timeMillis);  long timeConverted = t.TimeMillis;  Assert.AreEqual(timeMillis, timeConverted);  } |
| ***Salida*** | Nombre de la prueba: *TestTimeSeconds*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:02,8221745 |

### Serializador

#### Pruebas del BuiltinTopic

|  |  |
| --- | --- |
| **Llamada:**  public static org.omg.dds.type.typeobject.Type ExploreType(System.Type type) | |
| ***Descripción*** | En esta prueba se verifica el correcto funcionamiento del serializador de DDS en el Builtin Data Participant |
| ***Entrada*** | Inicialmente no se tiene inicializado al ddsType |
| ***Referencia*** |  |
| ***Código*** | [TestMethod]  public void TestParticipantBuiltinTopicData()  {  var ddsType = TypeExplorer.ExploreType(typeof(ParticipantBuiltinTopicData));  Assert.IsNotNull(ddsType);  Assert.IsNotNull(ddsType.GetProperty());  var propInfo = ddsType.GetProperty();  Assert.AreEqual("org.omg.dds.topic.ParticipantBuiltinTopicData", propInfo.Name);  Assert.IsInstanceOfType(ddsType, typeof(StructureType));  StructureType structType = ddsType as StructureType;  var members = structType.GetMember();  Assert.IsNotNull(members);  Assert.AreEqual(2, members.Count);  Assert.AreEqual("Key", members[0].GetProperty().Name);  Assert.AreEqual("UserData", members[1].GetProperty().Name);  Assert.AreEqual((uint)0x0050, members[0].GetProperty().MemberId);  Assert.AreEqual((uint)0x002C, members[1].GetProperty().MemberId);  } |
| ***Salida*** | Nombre de la prueba: *TestParticipantBuiltinTopicData*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00,1871316 |

|  |  |
| --- | --- |
| **Llamada:**  public static org.omg.dds.type.typeobject.Type ExploreType(System.Type type) | |
| ***Descripción*** | En esta prueba se verifica el correcto funcionamiento del serializador de DDS en el Builtin Data Publication |
| ***Entrada*** | Inicialmente no se tiene inicializado al ddsType |
| ***Referencia*** |  |
| ***Código*** | [TestMethod]  public void TestPublicationBuiltinTopicData()  {  var ddsType = TypeExplorer.ExploreType(typeof(PublicationBuiltinTopicData));  Assert.IsNotNull(ddsType);  Assert.IsNotNull(ddsType.GetProperty());  var propInfo = ddsType.GetProperty();  Assert.AreEqual("org.omg.dds.topic.PublicationBuiltinTopicData", propInfo.Name);  Assert.IsInstanceOfType(ddsType, typeof(StructureType));  StructureType structType = ddsType as StructureType;  var members = structType.GetMember();  Assert.IsNotNull(members);  Assert.AreEqual(24, members.Count);  Assert.AreEqual("Key", members[0].GetProperty().Name);  Assert.AreEqual("ParticipantKey", members[1].GetProperty().Name);  Assert.AreEqual("TopicName", members[2].GetProperty().Name);  Assert.AreEqual("TypeName", members[3].GetProperty().Name);  Assert.AreEqual("EquivalentTypeName", members[4].GetProperty().Name);  Assert.AreEqual("BaseTypeName", members[5].GetProperty().Name);  Assert.AreEqual("Type", members[6].GetProperty().Name);  Assert.AreEqual("Durability", members[7].GetProperty().Name);  Assert.AreEqual("DurabilityService", members[8].GetProperty().Name);  Assert.AreEqual("Deadline", members[9].GetProperty().Name);  Assert.AreEqual("LatencyBudget", members[10].GetProperty().Name);  Assert.AreEqual("Liveliness", members[11].GetProperty().Name);  Assert.AreEqual("Reliability", members[12].GetProperty().Name);  Assert.AreEqual("Lifespan", members[13].GetProperty().Name);  Assert.AreEqual("UserData", members[14].GetProperty().Name);  Assert.AreEqual("Ownership", members[15].GetProperty().Name);  Assert.AreEqual("OwnershipStrength", members[16].GetProperty().Name);  Assert.AreEqual("DestinationOrder", members[17].GetProperty().Name);  Assert.AreEqual("Presentation", members[18].GetProperty().Name);  Assert.AreEqual("Partition", members[19].GetProperty().Name);  Assert.AreEqual("TopicData", members[20].GetProperty().Name);  Assert.AreEqual("GroupData", members[21].GetProperty().Name);  Assert.AreEqual("Representation", members[22].GetProperty().Name);  Assert.AreEqual("TypeConsistency", members[23].GetProperty().Name);  Assert.AreEqual((uint)0x005A, members[0].GetProperty().MemberId);  Assert.AreEqual((uint)0x0050, members[1].GetProperty().MemberId);  Assert.AreEqual((uint)0x0005, members[2].GetProperty().MemberId);  Assert.AreEqual((uint)0x0007, members[3].GetProperty().MemberId);  Assert.AreEqual((uint)0x0075, members[4].GetProperty().MemberId);  Assert.AreEqual((uint)0x0076, members[5].GetProperty().MemberId);  Assert.AreEqual((uint)0x0072, members[6].GetProperty().MemberId);  Assert.AreEqual((uint)0x001D, members[7].GetProperty().MemberId);  Assert.AreEqual((uint)0x001E, members[8].GetProperty().MemberId);  Assert.AreEqual((uint)0x0023, members[9].GetProperty().MemberId);  Assert.AreEqual((uint)0x0027, members[10].GetProperty().MemberId);  Assert.AreEqual((uint)0x001B, members[11].GetProperty().MemberId);  Assert.AreEqual((uint)0x001A, members[12].GetProperty().MemberId);  Assert.AreEqual((uint)0x002B, members[13].GetProperty().MemberId);  Assert.AreEqual((uint)0x002C, members[14].GetProperty().MemberId);  Assert.AreEqual((uint)0x001F, members[15].GetProperty().MemberId);  Assert.AreEqual((uint)0x0006, members[16].GetProperty().MemberId);  Assert.AreEqual((uint)0x0025, members[17].GetProperty().MemberId);  Assert.AreEqual((uint)0x0021, members[18].GetProperty().MemberId);  Assert.AreEqual((uint)0x0029, members[19].GetProperty().MemberId);  Assert.AreEqual((uint)0x002E, members[20].GetProperty().MemberId);  Assert.AreEqual((uint)0x002D, members[21].GetProperty().MemberId);  Assert.AreEqual((uint)0x0073, members[22].GetProperty().MemberId);  Assert.AreEqual((uint)0x0074, members[23].GetProperty().MemberId);  } |
| ***Salida*** | Nombre de la prueba: *TestPublicationBuiltinTopicData*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00,1681182 |

|  |  |
| --- | --- |
| **Llamada:**  public static org.omg.dds.type.typeobject.Type ExploreType(System.Type type) | |
| ***Descripción*** | En esta prueba se verifica el correcto funcionamiento del serializador de DDS en el Builtin Data Suscriber |
| ***Entrada*** | Inicialmente no se tiene inicializado al ddsType |
| ***Referencia*** |  |
| ***Código*** | [TestMethod]  public void TestSubscriptionBuiltinTopicData()  {  var ddsType = TypeExplorer.ExploreType(typeof(SubscriptionBuiltinTopicData));  Assert.IsNotNull(ddsType);  Assert.IsNotNull(ddsType.GetProperty());  var propInfo = ddsType.GetProperty();  Assert.AreEqual("org.omg.dds.topic.SubscriptionBuiltinTopicData", propInfo.Name);  Assert.IsInstanceOfType(ddsType, typeof(StructureType));  StructureType structType = ddsType as StructureType;  var members = structType.GetMember();  Assert.IsNotNull(members);  Assert.AreEqual(22, members.Count);  Assert.AreEqual("Key", members[0].GetProperty().Name);  Assert.AreEqual("ParticipantKey", members[1].GetProperty().Name);  Assert.AreEqual("TopicName", members[2].GetProperty().Name);  Assert.AreEqual("TypeName", members[3].GetProperty().Name);  Assert.AreEqual("EquivalentTypeName", members[4].GetProperty().Name);  Assert.AreEqual("BaseTypeName", members[5].GetProperty().Name);  Assert.AreEqual("Type", members[6].GetProperty().Name);  Assert.AreEqual("Durability", members[7].GetProperty().Name);  Assert.AreEqual("Deadline", members[8].GetProperty().Name);  Assert.AreEqual("LatencyBudget", members[9].GetProperty().Name);  Assert.AreEqual("Liveliness", members[10].GetProperty().Name);  Assert.AreEqual("Reliability", members[11].GetProperty().Name);  Assert.AreEqual("Ownership", members[12].GetProperty().Name);  Assert.AreEqual("DestinationOrder", members[13].GetProperty().Name);  Assert.AreEqual("UserData", members[14].GetProperty().Name);  Assert.AreEqual("TimeBasedFilter", members[15].GetProperty().Name);  Assert.AreEqual("Presentation", members[16].GetProperty().Name);  Assert.AreEqual("Partition", members[17].GetProperty().Name);  Assert.AreEqual("TopicData", members[18].GetProperty().Name);  Assert.AreEqual("GroupData", members[19].GetProperty().Name);  Assert.AreEqual("Representation", members[20].GetProperty().Name);  Assert.AreEqual("TypeConsistency", members[21].GetProperty().Name);  Assert.AreEqual((uint)0x005A, members[0].GetProperty().MemberId);  Assert.AreEqual((uint)0x0050, members[1].GetProperty().MemberId);  Assert.AreEqual((uint)0x0005, members[2].GetProperty().MemberId);  Assert.AreEqual((uint)0x0007, members[3].GetProperty().MemberId);  Assert.AreEqual((uint)0x0075, members[4].GetProperty().MemberId);  Assert.AreEqual((uint)0x0076, members[5].GetProperty().MemberId);  Assert.AreEqual((uint)0x0072, members[6].GetProperty().MemberId);  Assert.AreEqual((uint)0x001D, members[7].GetProperty().MemberId);  Assert.AreEqual((uint)0x0023, members[8].GetProperty().MemberId);  Assert.AreEqual((uint)0x0027, members[9].GetProperty().MemberId);  Assert.AreEqual((uint)0x001B, members[10].GetProperty().MemberId);  Assert.AreEqual((uint)0x001A, members[11].GetProperty().MemberId);  Assert.AreEqual((uint)0x001F, members[12].GetProperty().MemberId);  Assert.AreEqual((uint)0x0025, members[13].GetProperty().MemberId);  Assert.AreEqual((uint)0x002C, members[14].GetProperty().MemberId);  Assert.AreEqual((uint)0x0004, members[15].GetProperty().MemberId);  Assert.AreEqual((uint)0x0021, members[16].GetProperty().MemberId);  Assert.AreEqual((uint)0x0029, members[17].GetProperty().MemberId);  Assert.AreEqual((uint)0x002E, members[18].GetProperty().MemberId);  Assert.AreEqual((uint)0x002D, members[19].GetProperty().MemberId);  Assert.AreEqual((uint)0x0073, members[20].GetProperty().MemberId);  Assert.AreEqual((uint)0x0074, members[21].GetProperty().MemberId);  } |
| ***Salida*** | Nombre de la prueba: *TestSubscriptionBuiltinTopicData*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00,1438964 |

|  |  |
| --- | --- |
| **Llamada:**  public static org.omg.dds.type.typeobject.Type ExploreType(System.Type type) | |
| ***Descripción*** | En esta prueba se verifica el correcto funcionamiento del serializador de DDS en el Builtin Data Suscriber |
| ***Entrada*** | Inicialmente no se tiene inicializado al ddsType |
| ***Referencia*** |  |
| ***Código*** | [TestMethod]  public void TestSubscriptionBuiltinTopicData()  {  var ddsType = TypeExplorer.ExploreType(typeof(SubscriptionBuiltinTopicData));  Assert.IsNotNull(ddsType);  Assert.IsNotNull(ddsType.GetProperty());  var propInfo = ddsType.GetProperty();  Assert.AreEqual("org.omg.dds.topic.SubscriptionBuiltinTopicData", propInfo.Name);  Assert.IsInstanceOfType(ddsType, typeof(StructureType));  StructureType structType = ddsType as StructureType;  var members = structType.GetMember();  Assert.IsNotNull(members);  Assert.AreEqual(22, members.Count);  Assert.AreEqual("Key", members[0].GetProperty().Name);  Assert.AreEqual("ParticipantKey", members[1].GetProperty().Name);  Assert.AreEqual("TopicName", members[2].GetProperty().Name);  Assert.AreEqual("TypeName", members[3].GetProperty().Name);  Assert.AreEqual("EquivalentTypeName", members[4].GetProperty().Name);  Assert.AreEqual("BaseTypeName", members[5].GetProperty().Name);  Assert.AreEqual("Type", members[6].GetProperty().Name);  Assert.AreEqual("Durability", members[7].GetProperty().Name);  Assert.AreEqual("Deadline", members[8].GetProperty().Name);  Assert.AreEqual("LatencyBudget", members[9].GetProperty().Name);  Assert.AreEqual("Liveliness", members[10].GetProperty().Name);  Assert.AreEqual("Reliability", members[11].GetProperty().Name);  Assert.AreEqual("Ownership", members[12].GetProperty().Name);  Assert.AreEqual("DestinationOrder", members[13].GetProperty().Name);  Assert.AreEqual("UserData", members[14].GetProperty().Name);  Assert.AreEqual("TimeBasedFilter", members[15].GetProperty().Name);  Assert.AreEqual("Presentation", members[16].GetProperty().Name);  Assert.AreEqual("Partition", members[17].GetProperty().Name);  Assert.AreEqual("TopicData", members[18].GetProperty().Name);  Assert.AreEqual("GroupData", members[19].GetProperty().Name);  Assert.AreEqual("Representation", members[20].GetProperty().Name);  Assert.AreEqual("TypeConsistency", members[21].GetProperty().Name);  Assert.AreEqual((uint)0x005A, members[0].GetProperty().MemberId);  Assert.AreEqual((uint)0x0050, members[1].GetProperty().MemberId);  Assert.AreEqual((uint)0x0005, members[2].GetProperty().MemberId);  Assert.AreEqual((uint)0x0007, members[3].GetProperty().MemberId);  Assert.AreEqual((uint)0x0075, members[4].GetProperty().MemberId);  Assert.AreEqual((uint)0x0076, members[5].GetProperty().MemberId);  Assert.AreEqual((uint)0x0072, members[6].GetProperty().MemberId);  Assert.AreEqual((uint)0x001D, members[7].GetProperty().MemberId);  Assert.AreEqual((uint)0x0023, members[8].GetProperty().MemberId);  Assert.AreEqual((uint)0x0027, members[9].GetProperty().MemberId);  Assert.AreEqual((uint)0x001B, members[10].GetProperty().MemberId);  Assert.AreEqual((uint)0x001A, members[11].GetProperty().MemberId);  Assert.AreEqual((uint)0x001F, members[12].GetProperty().MemberId);  Assert.AreEqual((uint)0x0025, members[13].GetProperty().MemberId);  Assert.AreEqual((uint)0x002C, members[14].GetProperty().MemberId);  Assert.AreEqual((uint)0x0004, members[15].GetProperty().MemberId);  Assert.AreEqual((uint)0x0021, members[16].GetProperty().MemberId);  Assert.AreEqual((uint)0x0029, members[17].GetProperty().MemberId);  Assert.AreEqual((uint)0x002E, members[18].GetProperty().MemberId);  Assert.AreEqual((uint)0x002D, members[19].GetProperty().MemberId);  Assert.AreEqual((uint)0x0073, members[20].GetProperty().MemberId);  Assert.AreEqual((uint)0x0074, members[21].GetProperty().MemberId);  } |
| ***Salida*** | Nombre de la prueba: *TestSubscriptionBuiltinTopicData*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00,1438964 |

|  |  |
| --- | --- |
| **Llamada:**  public static org.omg.dds.type.typeobject.Type ExploreType(System.Type type) | |
| ***Descripción*** | En esta prueba se verifica el correcto funcionamiento del serializador de DDS en el Builtin Data Topic |
| ***Entrada*** | Inicialmente no se tiene inicializado al ddsType |
| ***Referencia*** |  |
| ***Código*** | [TestMethod]  public void TestTopicBuiltinTopicData()  {  var ddsType = TypeExplorer.ExploreType(typeof(TopicBuiltinTopicData));  Assert.IsNotNull(ddsType);  Assert.IsNotNull(ddsType.GetProperty());  var propInfo = ddsType.GetProperty();  Assert.AreEqual("org.omg.dds.topic.TopicBuiltinTopicData", propInfo.Name);  Assert.IsInstanceOfType(ddsType, typeof(StructureType));  StructureType structType = ddsType as StructureType;  var members = structType.GetMember();  Assert.IsNotNull(members);  Assert.AreEqual(21, members.Count);  Assert.AreEqual("Key", members[0].GetProperty().Name);  Assert.AreEqual("Name", members[1].GetProperty().Name);  Assert.AreEqual("TypeName", members[2].GetProperty().Name);  Assert.AreEqual("EquivalentTypeName", members[3].GetProperty().Name);  Assert.AreEqual("BaseTypeName", members[4].GetProperty().Name);  Assert.AreEqual("Type", members[5].GetProperty().Name);  Assert.AreEqual("Durability", members[6].GetProperty().Name);  Assert.AreEqual("DurabilityService", members[7].GetProperty().Name);  Assert.AreEqual("Deadline", members[8].GetProperty().Name);  Assert.AreEqual("LatencyBudget", members[9].GetProperty().Name);  Assert.AreEqual("Liveliness", members[10].GetProperty().Name);  Assert.AreEqual("Reliability", members[11].GetProperty().Name);  Assert.AreEqual("TransportPriority", members[12].GetProperty().Name);  Assert.AreEqual("Lifespan", members[13].GetProperty().Name);  Assert.AreEqual("DestinationOrder", members[14].GetProperty().Name);  Assert.AreEqual("History", members[15].GetProperty().Name);  Assert.AreEqual("ResourceLimits", members[16].GetProperty().Name);  Assert.AreEqual("Ownership", members[17].GetProperty().Name);  Assert.AreEqual("TopicData", members[18].GetProperty().Name);  Assert.AreEqual("Representation", members[19].GetProperty().Name);  Assert.AreEqual("TypeConsistency", members[20].GetProperty().Name);  Assert.AreEqual((uint)0x005A, members[0].GetProperty().MemberId);  Assert.AreEqual((uint)0x0005, members[1].GetProperty().MemberId);  Assert.AreEqual((uint)0x0007, members[2].GetProperty().MemberId);  Assert.AreEqual((uint)0x0075, members[3].GetProperty().MemberId);  Assert.AreEqual((uint)0x0076, members[4].GetProperty().MemberId);  Assert.AreEqual((uint)0x0072, members[5].GetProperty().MemberId);  Assert.AreEqual((uint)0x001D, members[6].GetProperty().MemberId);  Assert.AreEqual((uint)0x001E, members[7].GetProperty().MemberId);  Assert.AreEqual((uint)0x0023, members[8].GetProperty().MemberId);  Assert.AreEqual((uint)0x0027, members[9].GetProperty().MemberId);  Assert.AreEqual((uint)0x001B, members[10].GetProperty().MemberId);  Assert.AreEqual((uint)0x001A, members[11].GetProperty().MemberId);  Assert.AreEqual((uint)0x0049, members[12].GetProperty().MemberId);  Assert.AreEqual((uint)0x002B, members[13].GetProperty().MemberId);  Assert.AreEqual((uint)0x0025, members[14].GetProperty().MemberId);  Assert.AreEqual((uint)0x0040, members[15].GetProperty().MemberId);  Assert.AreEqual((uint)0x0041, members[16].GetProperty().MemberId);  Assert.AreEqual((uint)0x001F, members[17].GetProperty().MemberId);  Assert.AreEqual((uint)0x002E, members[18].GetProperty().MemberId);  Assert.AreEqual((uint)0x0073, members[19].GetProperty().MemberId);  Assert.AreEqual((uint)0x0074, members[20].GetProperty().MemberId);  } |
| ***Salida*** | Nombre de la prueba: *TestTopicBuiltinTopicData*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00,1909715 |

#### Pruebas de Encapsulación CDR.

|  |  |
| --- | --- |
| **Llamada:**  public CDREncapsulation(IoBuffer buffer, object dataObj, ByteOrder order)  public static void Serialize(IoBuffer buffer, object dataObj, ByteOrder order)  public static T Deserialize<T>(IoBuffer buffer) | |
| ***Descripción*** | En esta prueba se muestra el corrector funcionamiento del CDREncapsulation para el BoolPacket Little Endian |
| ***Entrada*** | Dentro del Test Initialize se inicializa el serializador |
| ***Referencia*** |  |
| ***Código*** | [TestMethod]  public void TestBoolPacketLE()  {  BoolPacket v1 = new BoolPacket(true);  int bufferSize = sizeof(bool) + CDRHeaderSize;  var buffer = ByteBufferAllocator.Instance.Allocate(bufferSize);  CDREncapsulation.Serialize(buffer, v1, ByteOrder.LittleEndian);  Assert.AreEqual(bufferSize, buffer.Position);  buffer.Rewind();  Assert.AreEqual("00 01 00 00 01", buffer.GetHexDump());  BoolPacket v2 = CDREncapsulation.Deserialize<BoolPacket>(buffer);  Assert.AreEqual(v1, v2);  Assert.AreEqual(bufferSize, buffer.Position);  } |
| ***Salida*** | Nombre de la prueba: *TestBoolPacketLE*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00,1651349 |

|  |  |
| --- | --- |
| **Llamada:**  public CDREncapsulation(IoBuffer buffer, object dataObj, ByteOrder order)  public static void Serialize(IoBuffer buffer, object dataObj, ByteOrder order)  public static T Deserialize<T>(IoBuffer buffer) | |
| ***Descripción*** | En esta prueba se muestra el corrector funcionamiento del CDREncapsulation para el CharPacket Little Endian |
| ***Entrada*** | Dentro del Test Initialize se inicializa el serializador |
| ***Referencia*** |  |
| ***Código*** | [TestMethod]  public void TestCharPacketLE()  {  CharPacket v1 = new CharPacket('A');  int bufferSize = sizeof(char) + CDRHeaderSize;  var buffer = ByteBufferAllocator.Instance.Allocate(bufferSize);  CDREncapsulation.Serialize(buffer, v1, ByteOrder.LittleEndian);  Assert.AreEqual(bufferSize, buffer.Position);  buffer.Rewind();  Assert.AreEqual("00 01 00 00 41 00", buffer.GetHexDump());  CharPacket v2 = CDREncapsulation.Deserialize<CharPacket>(buffer);  Assert.AreEqual(v1, v2);  Assert.AreEqual(bufferSize, buffer.Position);  } |
| ***Salida*** | Nombre de la prueba: *TestCharPacketLE*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00,2107132 |

|  |  |
| --- | --- |
| **Llamada:**  public CDREncapsulation(IoBuffer buffer, object dataObj, ByteOrder order)  public static void Serialize(IoBuffer buffer, object dataObj, ByteOrder order)  public static T Deserialize<T>(IoBuffer buffer) | |
| ***Descripción*** | En esta prueba se muestra el corrector funcionamiento del CDREncapsulation para el U8Packet Little Endian |
| ***Entrada*** | Dentro del Test Initialize se inicializa el serializador |
| ***Referencia*** |  |
| ***Código*** | [TestMethod]  public void TestU8PacketLE()  {  U8Packet v1 = new U8Packet(0xA);  int bufferSize = sizeof(byte) + CDRHeaderSize;  var buffer = ByteBufferAllocator.Instance.Allocate(bufferSize);  CDREncapsulation.Serialize(buffer, v1, ByteOrder.LittleEndian);  Assert.AreEqual(bufferSize, buffer.Position);  buffer.Rewind();  Assert.AreEqual("00 01 00 00 0A", buffer.GetHexDump());  U8Packet v2 = CDREncapsulation.Deserialize<U8Packet>(buffer);  Assert.AreEqual(v1, v2);  Assert.AreEqual(bufferSize, buffer.Position);  } |
| ***Salida*** | Nombre de la prueba: *TestU8PacketLE*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00,2791979 |

|  |  |
| --- | --- |
| **Llamada:**  public CDREncapsulation(IoBuffer buffer, object dataObj, ByteOrder order)  public static void Serialize(IoBuffer buffer, object dataObj, ByteOrder order)  public static T Deserialize<T>(IoBuffer buffer) | |
| ***Descripción*** | En esta prueba se muestra el corrector funcionamiento del CDREncapsulation para el U16Packet Little Endian |
| ***Entrada*** | Dentro del Test Initialize se inicializa el serializador |
| ***Referencia*** |  |
| ***Código*** | [TestMethod]  public void TestU16PacketLE()  {  U16Packet v1 = new U16Packet(0xAB);  int bufferSize = sizeof(ushort) + CDRHeaderSize;  var buffer = ByteBufferAllocator.Instance.Allocate(bufferSize);  CDREncapsulation.Serialize(buffer, v1, ByteOrder.LittleEndian);  Assert.AreEqual(bufferSize, buffer.Position);  buffer.Rewind();  Assert.AreEqual("00 01 00 00 AB 00", buffer.GetHexDump());  U16Packet v2 = CDREncapsulation.Deserialize<U16Packet>(buffer);  Assert.AreEqual(v1, v2);  Assert.AreEqual(bufferSize, buffer.Position);  } |
| ***Salida*** | Nombre de la prueba: *TestU16PacketLE*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00,1831424 |

|  |  |
| --- | --- |
| **Llamada:**  public CDREncapsulation(IoBuffer buffer, object dataObj, ByteOrder order)  public static void Serialize(IoBuffer buffer, object dataObj, ByteOrder order)  public static T Deserialize<T>(IoBuffer buffer) | |
| ***Descripción*** | En esta prueba se muestra el corrector funcionamiento del CDREncapsulation para el U32Packet Little Endian |
| ***Entrada*** | Dentro del Test Initialize se inicializa el serializador |
| ***Referencia*** |  |
| ***Código*** | [TestMethod]  public void TestU32PacketLE()  {  U32Packet v1 = new U32Packet(0xABA);  int bufferSize = sizeof(uint) + CDRHeaderSize;  var buffer = ByteBufferAllocator.Instance.Allocate(bufferSize);  CDREncapsulation.Serialize(buffer, v1, ByteOrder.LittleEndian);  Assert.AreEqual(bufferSize, buffer.Position);  buffer.Rewind();  Assert.AreEqual("00 01 00 00 BA 0A 00 00", buffer.GetHexDump());  U32Packet v2 = CDREncapsulation.Deserialize<U32Packet>(buffer);  Assert.AreEqual(v1, v2);  Assert.AreEqual(bufferSize, buffer.Position);  } |
| ***Salida*** | Nombre de la prueba: *TestU32PacketLE*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00,171796 |

|  |  |
| --- | --- |
| **Llamada:**  public CDREncapsulation(IoBuffer buffer, object dataObj, ByteOrder order)  public static void Serialize(IoBuffer buffer, object dataObj, ByteOrder order)  public static T Deserialize<T>(IoBuffer buffer) | |
| ***Descripción*** | En esta prueba se muestra el corrector funcionamiento del CDREncapsulation para el U64Packet Little Endian |
| ***Entrada*** | Dentro del Test Initialize se inicializa el serializador |
| ***Referencia*** |  |
| ***Código*** | [TestMethod]  public void TestU64PacketLE()  {  U64Packet v1 = new U64Packet(0xABCDEF);  int bufferSize = sizeof(ulong) + CDRHeaderSize;  var buffer = ByteBufferAllocator.Instance.Allocate(bufferSize);  CDREncapsulation.Serialize(buffer, v1, ByteOrder.LittleEndian);  Assert.AreEqual(bufferSize, buffer.Position);  buffer.Rewind();  Assert.AreEqual("00 01 00 00 EF CD AB 00 00 00 00 00", buffer.GetHexDump());  U64Packet v2 = CDREncapsulation.Deserialize<U64Packet>(buffer);  Assert.AreEqual(v1, v2);  Assert.AreEqual(bufferSize, buffer.Position);  } |
| ***Salida*** | Nombre de la prueba: *TestU64PacketLE*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00,1623851 |

|  |  |
| --- | --- |
| **Llamada:**  public CDREncapsulation(IoBuffer buffer, object dataObj, ByteOrder order)  public static void Serialize(IoBuffer buffer, object dataObj, ByteOrder order)  public static T Deserialize<T>(IoBuffer buffer) | |
| ***Descripción*** | En esta prueba se muestra el corrector funcionamiento del CDREncapsulation para el S8Packet Little Endian |
| ***Entrada*** | Dentro del Test Initialize se inicializa el serializador |
| ***Referencia*** |  |
| ***Código*** | [TestMethod]  public void TestS8PacketLE1()  {  S8Packet v1 = new S8Packet(-1);  int bufferSize = sizeof(sbyte) + CDRHeaderSize;  var buffer = ByteBufferAllocator.Instance.Allocate(bufferSize);  CDREncapsulation.Serialize(buffer, v1, ByteOrder.LittleEndian);  Assert.AreEqual(bufferSize, buffer.Position);  buffer.Rewind();  Assert.AreEqual("00 01 00 00 FF", buffer.GetHexDump());  S8Packet v2 = CDREncapsulation.Deserialize<S8Packet>(buffer);  Assert.AreEqual(v1, v2);  Assert.AreEqual(bufferSize, buffer.Position);  } |
| ***Salida*** | Nombre de la prueba: *TestS8PacketLE1*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00,1973871 |

|  |  |
| --- | --- |
| **Llamada:**  public CDREncapsulation(IoBuffer buffer, object dataObj, ByteOrder order)  public static void Serialize(IoBuffer buffer, object dataObj, ByteOrder order)  public static T Deserialize<T>(IoBuffer buffer) | |
| ***Descripción*** | En esta prueba se muestra el corrector funcionamiento del CDREncapsulation para el S8Packet Little Endian |
| ***Entrada*** | Dentro del Test Initialize se inicializa el serializador |
| ***Referencia*** |  |
| ***Código*** | [TestMethod]  public void TestS8PacketLE2()  {  S8Packet v1 = new S8Packet(+1);  int bufferSize = sizeof(sbyte) + CDRHeaderSize;  var buffer = ByteBufferAllocator.Instance.Allocate(bufferSize);  CDREncapsulation.Serialize(buffer, v1, ByteOrder.LittleEndian);  Assert.AreEqual(bufferSize, buffer.Position);  buffer.Rewind();  Assert.AreEqual("00 01 00 00 01", buffer.GetHexDump());  S8Packet v2 = CDREncapsulation.Deserialize<S8Packet>(buffer);  Assert.AreEqual(v1, v2);  Assert.AreEqual(bufferSize, buffer.Position);  } |
| ***Salida*** | Nombre de la prueba: *TestS8PacketLE2*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00,1755769 |

|  |  |
| --- | --- |
| **Llamada:**  public CDREncapsulation(IoBuffer buffer, object dataObj, ByteOrder order)  public static void Serialize(IoBuffer buffer, object dataObj, ByteOrder order)  public static T Deserialize<T>(IoBuffer buffer) | |
| ***Descripción*** | En esta prueba se muestra el corrector funcionamiento del CDREncapsulation para el S16Packet Little Endian |
| ***Entrada*** | Dentro del Test Initialize se inicializa el serializador |
| ***Referencia*** |  |
| ***Código*** | [TestMethod]  public void TestS16PacketLE1()  {  S16Packet v1 = new S16Packet(-10);  int bufferSize = sizeof(short) + CDRHeaderSize;  var buffer = ByteBufferAllocator.Instance.Allocate(bufferSize);  CDREncapsulation.Serialize(buffer, v1, ByteOrder.LittleEndian);  Assert.AreEqual(bufferSize, buffer.Position);  buffer.Rewind();  Assert.AreEqual("00 01 00 00 F6 FF", buffer.GetHexDump());  S16Packet v2 = CDREncapsulation.Deserialize<S16Packet>(buffer);  Assert.AreEqual(v1, v2);  Assert.AreEqual(bufferSize, buffer.Position);  } |
| ***Salida*** | Nombre de la prueba: *TestS16PacketLE1*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00,1916198 |

|  |  |
| --- | --- |
| **Llamada:**  public CDREncapsulation(IoBuffer buffer, object dataObj, ByteOrder order)  public static void Serialize(IoBuffer buffer, object dataObj, ByteOrder order)  public static T Deserialize<T>(IoBuffer buffer) | |
| ***Descripción*** | En esta prueba se muestra el corrector funcionamiento del CDREncapsulation para el S16Packet Little Endian |
| ***Entrada*** | Dentro del Test Initialize se inicializa el serializador |
| ***Referencia*** |  |
| ***Código*** | [TestMethod]  public void TestS16PacketLE2()  {  S16Packet v1 = new S16Packet(+10);  int bufferSize = sizeof(short) + CDRHeaderSize;  var buffer = ByteBufferAllocator.Instance.Allocate(bufferSize);  CDREncapsulation.Serialize(buffer, v1, ByteOrder.LittleEndian);  Assert.AreEqual(bufferSize, buffer.Position);  buffer.Rewind();  Assert.AreEqual("00 01 00 00 0A 00", buffer.GetHexDump());  S16Packet v2 = CDREncapsulation.Deserialize<S16Packet>(buffer);  Assert.AreEqual(v1, v2);  Assert.AreEqual(bufferSize, buffer.Position);  } |
| ***Salida*** | Nombre de la prueba: *TestS16PacketLE2*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00,2151697 |

|  |  |
| --- | --- |
| **Llamada:**  public CDREncapsulation(IoBuffer buffer, object dataObj, ByteOrder order)  public static void Serialize(IoBuffer buffer, object dataObj, ByteOrder order)  public static T Deserialize<T>(IoBuffer buffer) | |
| ***Descripción*** | En esta prueba se muestra el corrector funcionamiento del CDREncapsulation para el S32Packet Little Endian |
| ***Entrada*** | Dentro del Test Initialize se inicializa el serializador |
| ***Referencia*** |  |
| ***Código*** | [TestMethod]  public void TestS32PacketLE1()  {  S32Packet v1 = new S32Packet(-0xABA);  int bufferSize = sizeof(int) + CDRHeaderSize;  var buffer = ByteBufferAllocator.Instance.Allocate(bufferSize);  CDREncapsulation.Serialize(buffer, v1, ByteOrder.LittleEndian);  Assert.AreEqual(bufferSize, buffer.Position);  buffer.Rewind();  Assert.AreEqual("00 01 00 00 46 F5 FF FF", buffer.GetHexDump());  S32Packet v2 = CDREncapsulation.Deserialize<S32Packet>(buffer);  Assert.AreEqual(v1, v2);  Assert.AreEqual(bufferSize, buffer.Position);  } |
| ***Salida*** | Nombre de la prueba: *TestS32PacketLE1*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00,15711 |

|  |  |
| --- | --- |
| **Llamada:**  public CDREncapsulation(IoBuffer buffer, object dataObj, ByteOrder order)  public static void Serialize(IoBuffer buffer, object dataObj, ByteOrder order)  public static T Deserialize<T>(IoBuffer buffer) | |
| ***Descripción*** | En esta prueba se muestra el corrector funcionamiento del CDREncapsulation para el S32Packet Little Endian |
| ***Entrada*** | Dentro del Test Initialize se inicializa el serializador |
| ***Referencia*** |  |
| ***Código*** | [TestMethod]  public void TestS32PacketLE2()  {  S32Packet v1 = new S32Packet(0xABA);  int bufferSize = sizeof(int) + CDRHeaderSize;  var buffer = ByteBufferAllocator.Instance.Allocate(bufferSize);  CDREncapsulation.Serialize(buffer, v1, ByteOrder.LittleEndian);  Assert.AreEqual(bufferSize, buffer.Position);  buffer.Rewind();  Assert.AreEqual("00 01 00 00 BA 0A 00 00", buffer.GetHexDump());  S32Packet v2 = CDREncapsulation.Deserialize<S32Packet>(buffer);  Assert.AreEqual(v1, v2);  Assert.AreEqual(bufferSize, buffer.Position);  } |
| ***Salida*** | Nombre de la prueba: *TestS32PacketLE2*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00,1565519 |

|  |  |
| --- | --- |
| **Llamada:**  public CDREncapsulation(IoBuffer buffer, object dataObj, ByteOrder order)  public static void Serialize(IoBuffer buffer, object dataObj, ByteOrder order)  public static T Deserialize<T>(IoBuffer buffer) | |
| ***Descripción*** | En esta prueba se muestra el corrector funcionamiento del CDREncapsulation para el S4Packet Little Endian |
| ***Entrada*** | Dentro del Test Initialize se inicializa el serializador |
| ***Referencia*** |  |
| ***Código*** | [TestMethod]  public void TestS64PacketLE1()  {  S64Packet v1 = new S64Packet(-0xABCD);  int bufferSize = sizeof(long) + CDRHeaderSize;  var buffer = ByteBufferAllocator.Instance.Allocate(bufferSize);  CDREncapsulation.Serialize(buffer, v1, ByteOrder.LittleEndian);  Assert.AreEqual(bufferSize, buffer.Position);  buffer.Rewind();  Assert.AreEqual("00 01 00 00 33 54 FF FF FF FF FF FF", buffer.GetHexDump());  S64Packet v2 = CDREncapsulation.Deserialize<S64Packet>(buffer);  Assert.AreEqual(v1, v2);  Assert.AreEqual(bufferSize, buffer.Position);  } |
| ***Salida*** | Nombre de la prueba: *TestS64PacketLE1*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00,1781261 |

|  |  |
| --- | --- |
| **Llamada:**  public CDREncapsulation(IoBuffer buffer, object dataObj, ByteOrder order)  public static void Serialize(IoBuffer buffer, object dataObj, ByteOrder order)  public static T Deserialize<T>(IoBuffer buffer) | |
| ***Descripción*** | En esta prueba se muestra el corrector funcionamiento del CDREncapsulation para el S4Packet Little Endian |
| ***Entrada*** | Dentro del Test Initialize se inicializa el serializador |
| ***Referencia*** |  |
| ***Código*** | [TestMethod]  public void TestS64PacketLE2()  {  S64Packet v1 = new S64Packet(0xABCD);  int bufferSize = sizeof(long) + CDRHeaderSize;  var buffer = ByteBufferAllocator.Instance.Allocate(bufferSize);  CDREncapsulation.Serialize(buffer, v1, ByteOrder.LittleEndian);  Assert.AreEqual(bufferSize, buffer.Position);  buffer.Rewind();  Assert.AreEqual("00 01 00 00 CD AB 00 00 00 00 00 00", buffer.GetHexDump());  S64Packet v2 = CDREncapsulation.Deserialize<S64Packet>(buffer);  Assert.AreEqual(v1, v2);  Assert.AreEqual(bufferSize, buffer.Position);  } |
| ***Salida*** | Nombre de la prueba: *TestS64PacketLE2*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00,1868797 |

|  |  |
| --- | --- |
| **Llamada:**  public CDREncapsulation(IoBuffer buffer, object dataObj, ByteOrder order)  public static void Serialize(IoBuffer buffer, object dataObj, ByteOrder order)  public static T Deserialize<T>(IoBuffer buffer) | |
| ***Descripción*** | En esta prueba se muestra el corrector funcionamiento del CDREncapsulation para el SinglePacket Little Endian |
| ***Entrada*** | Dentro del Test Initialize se inicializa el serializador |
| ***Referencia*** |  |
| ***Código*** | [TestMethod]  public void TestSinglePacketLE()  {  SinglePacket v1 = new SinglePacket(0.1f);  int bufferSize = sizeof(float) + CDRHeaderSize;  var buffer = ByteBufferAllocator.Instance.Allocate(bufferSize);  CDREncapsulation.Serialize(buffer, v1, ByteOrder.LittleEndian);  Assert.AreEqual(bufferSize, buffer.Position);  buffer.Rewind();  Assert.AreEqual("00 01 00 00 CD CC CC 3D", buffer.GetHexDump());  SinglePacket v2 = CDREncapsulation.Deserialize<SinglePacket>(buffer);  Assert.AreEqual(v1, v2);  Assert.AreEqual(bufferSize, buffer.Position);  } |
| ***Salida*** | Nombre de la prueba: *TestSinglePacketLE*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00,1841175 |

|  |  |
| --- | --- |
| **Llamada:**  public CDREncapsulation(IoBuffer buffer, object dataObj, ByteOrder order)  public static void Serialize(IoBuffer buffer, object dataObj, ByteOrder order)  public static T Deserialize<T>(IoBuffer buffer) | |
| ***Descripción*** | En esta prueba se muestra el corrector funcionamiento del CDREncapsulation para el DoublePacket Little Endian |
| ***Entrada*** | Dentro del Test Initialize se inicializa el serializador |
| ***Referencia*** |  |
| ***Código*** | [TestMethod]  public void TestDoublePacketLE()  {  DoublePacket v1 = new DoublePacket(0.1);  int bufferSize = sizeof(double) + CDRHeaderSize;  var buffer = ByteBufferAllocator.Instance.Allocate(bufferSize);  CDREncapsulation.Serialize(buffer, v1, ByteOrder.LittleEndian);  Assert.AreEqual(bufferSize, buffer.Position);  buffer.Rewind();  Assert.AreEqual("00 01 00 00 9A 99 99 99 99 99 B9 3F", buffer.GetHexDump());  DoublePacket v2 = CDREncapsulation.Deserialize<DoublePacket>(buffer);  Assert.AreEqual(v1, v2);  Assert.AreEqual(bufferSize, buffer.Position);  } |
| ***Salida*** | Nombre de la prueba: *TestDoublePacketLE*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00,1841175 |

|  |  |
| --- | --- |
| **Llamada:**  public CDREncapsulation(IoBuffer buffer, object dataObj, ByteOrder order)  public static void Serialize(IoBuffer buffer, object dataObj, ByteOrder order)  public static T Deserialize<T>(IoBuffer buffer) | |
| ***Descripción*** | En esta prueba se muestra el corrector funcionamiento del CDREncapsulation para el BoolSequencePacket Little Endian |
| ***Entrada*** | Dentro del Test Initialize se inicializa el serializador |
| ***Referencia*** |  |
| ***Código*** | [TestMethod]  public void TestBoolSequencePacketLE()  {  BoolSequencePacket v1 = new BoolSequencePacket(new bool[] { true, false, false, true });  int bufferSize = v1.Size + ArrayHeader + CDRHeaderSize;  var buffer = ByteBufferAllocator.Instance.Allocate(bufferSize);  CDREncapsulation.Serialize(buffer, v1, ByteOrder.LittleEndian);  Assert.AreEqual(bufferSize, buffer.Position);  buffer.Rewind();  Assert.AreEqual("00 01 00 00 04 00 00 00 01 00 00 01", buffer.GetHexDump());  BoolSequencePacket v2 = CDREncapsulation.Deserialize<BoolSequencePacket>(buffer);  Assert.AreEqual(v1, v2);  Assert.AreEqual(bufferSize, buffer.Position);  } |
| ***Salida*** | Nombre de la prueba: *TestBoolSequencePacketLE*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00,1841175 |

|  |  |
| --- | --- |
| **Llamada:**  public CDREncapsulation(IoBuffer buffer, object dataObj, ByteOrder order)  public static void Serialize(IoBuffer buffer, object dataObj, ByteOrder order)  public static T Deserialize<T>(IoBuffer buffer) | |
| ***Descripción*** | En esta prueba se muestra el corrector funcionamiento del CDREncapsulation para el ShortSequencePacket Little Endian |
| ***Entrada*** | Dentro del Test Initialize se inicializa el serializador |
| ***Referencia*** |  |
| ***Código*** | [TestMethod]  public void TestShortSequencePacketLE()  {  ShortSequencePacket v1 = new ShortSequencePacket(new short[] { 0xFA1, 0xFF0, 0xB2F, 0x001 });  int bufferSize = v1.Size + ArrayHeader + CDRHeaderSize;  var buffer = ByteBufferAllocator.Instance.Allocate(bufferSize);  CDREncapsulation.Serialize(buffer, v1, ByteOrder.LittleEndian);  Assert.AreEqual(bufferSize, buffer.Position);  buffer.Rewind();  Assert.AreEqual("00 01 00 00 04 00 00 00 A1 0F F0 0F 2F 0B 01 00", buffer.GetHexDump());  ShortSequencePacket v2 = CDREncapsulation.Deserialize<ShortSequencePacket>(buffer);  Assert.AreEqual(v1, v2);  Assert.AreEqual(bufferSize, buffer.Position);  } |
| ***Salida*** | Nombre de la prueba: *TestShortSequencePacketLE*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00,1904143 |

|  |  |
| --- | --- |
| **Llamada:**  public CDREncapsulation(IoBuffer buffer, object dataObj, ByteOrder order)  public static void Serialize(IoBuffer buffer, object dataObj, ByteOrder order)  public static T Deserialize<T>(IoBuffer buffer) | |
| ***Descripción*** | En esta prueba se muestra el corrector funcionamiento del CDREncapsulation para el EnumSequencePacket Little Endian |
| ***Entrada*** | Dentro del Test Initialize se inicializa el serializador |
| ***Referencia*** |  |
| ***Código*** | [TestMethod]  public void TestEnumSequencePacketLE()  {  EnumSequencePacket v1 = new EnumSequencePacket(new MyEnum[] { MyEnum.Four, MyEnum.Three, MyEnum.Three, MyEnum.Zero, MyEnum.One, MyEnum.Five });  int bufferSize = v1.Size + ArrayHeader + CDRHeaderSize;  var buffer = ByteBufferAllocator.Instance.Allocate(bufferSize);  CDREncapsulation.Serialize(buffer, v1, ByteOrder.LittleEndian);  Assert.AreEqual(bufferSize, buffer.Position);  buffer.Rewind();  Assert.AreEqual("00 01 00 00 06 00 00 00 04 00 00 00 03 00 00 00 03 00 00 00 00 00 00 00 01 00 00 00 05 00 00 00", buffer.GetHexDump());  EnumSequencePacket v2 = CDREncapsulation.Deserialize<EnumSequencePacket>(buffer);  Assert.AreEqual(v1, v2);  Assert.AreEqual(bufferSize, buffer.Position);  } |
| ***Salida*** | Nombre de la prueba: *TestEnumSequencePacketLE*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00,1877371 |

|  |  |
| --- | --- |
| **Llamada:**  public CDREncapsulation(IoBuffer buffer, object dataObj, ByteOrder order)  public static void Serialize(IoBuffer buffer, object dataObj, ByteOrder order)  public static T Deserialize<T>(IoBuffer buffer) | |
| ***Descripción*** | En esta prueba se muestra el corrector funcionamiento del CDREncapsulation para el IntSequencePacket Little Endian |
| ***Entrada*** | Dentro del Test Initialize se inicializa el serializador |
| ***Referencia*** |  |
| ***Código*** | [TestMethod]  public void TestIntSequencePacketLE()  {  IntSequencePacket v1 = new IntSequencePacket(new int[] { 0xFFA1F0, 0xFF230F, 0xB2000F, 0xFFFFF01 });  int bufferSize = v1.Size + ArrayHeader + CDRHeaderSize;  var buffer = ByteBufferAllocator.Instance.Allocate(bufferSize);  CDREncapsulation.Serialize(buffer, v1, ByteOrder.LittleEndian);  Assert.AreEqual(bufferSize, buffer.Position);  buffer.Rewind();  Assert.AreEqual("00 01 00 00 04 00 00 00 F0 A1 FF 00 0F 23 FF 00 0F 00 B2 00 01 FF FF 0F", buffer.GetHexDump());  IntSequencePacket v2 = CDREncapsulation.Deserialize<IntSequencePacket>(buffer);  Assert.AreEqual(v1, v2);  Assert.AreEqual(bufferSize, buffer.Position);  } |
| ***Salida*** | Nombre de la prueba: *TestIntSequencePacketLE*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00,1932013 |

|  |  |
| --- | --- |
| **Llamada:**  public CDREncapsulation(IoBuffer buffer, object dataObj, ByteOrder order)  public static void Serialize(IoBuffer buffer, object dataObj, ByteOrder order)  public static T Deserialize<T>(IoBuffer buffer) | |
| ***Descripción*** | En esta prueba se muestra el corrector funcionamiento del CDREncapsulation para el EmptyBoolSequencePacket Little Endian |
| ***Entrada*** | Dentro del Test Initialize se inicializa el serializador |
| ***Referencia*** |  |
| ***Código*** | [TestMethod]  public void TestEmptyBoolSequencePacketLE()  {  BoolSequencePacket v1 = new BoolSequencePacket(new bool[] { });  int bufferSize = v1.Size + ArrayHeader + CDRHeaderSize;  var buffer = ByteBufferAllocator.Instance.Allocate(bufferSize);  CDREncapsulation.Serialize(buffer, v1, ByteOrder.LittleEndian);  Assert.AreEqual(bufferSize, buffer.Position);  buffer.Rewind();  Assert.AreEqual("00 01 00 00 00 00 00 00", buffer.GetHexDump());  BoolSequencePacket v2 = CDREncapsulation.Deserialize<BoolSequencePacket>(buffer);  Assert.AreEqual(v1, v2);  Assert.AreEqual(bufferSize, buffer.Position);  } |
| ***Salida*** | Nombre de la prueba: *TestEmptyBoolSequencePacketLE*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00,1851345 |

|  |  |
| --- | --- |
| **Llamada:**  public CDREncapsulation(IoBuffer buffer, object dataObj, ByteOrder order)  public static void Serialize(IoBuffer buffer, object dataObj, ByteOrder order)  public static T Deserialize<T>(IoBuffer buffer) | |
| ***Descripción*** | En esta prueba se muestra el corrector funcionamiento del CDREncapsulation para el EmptyShortSequencePacket Little Endian |
| ***Entrada*** | Dentro del Test Initialize se inicializa el serializador |
| ***Referencia*** |  |
| ***Código*** | [TestMethod]  public void TestEmptyShortSequencePacketLE()  {  ShortSequencePacket v1 = new ShortSequencePacket(new short[] { });  int bufferSize = v1.Size + ArrayHeader + CDRHeaderSize;  var buffer = ByteBufferAllocator.Instance.Allocate(bufferSize);  CDREncapsulation.Serialize(buffer, v1, ByteOrder.LittleEndian);  Assert.AreEqual(bufferSize, buffer.Position);  buffer.Rewind();  Assert.AreEqual("00 01 00 00 00 00 00 00", buffer.GetHexDump());  ShortSequencePacket v2 = CDREncapsulation.Deserialize<ShortSequencePacket>(buffer);  Assert.AreEqual(v1, v2);  Assert.AreEqual(bufferSize, buffer.Position);  } |
| ***Salida*** | Nombre de la prueba: *TestEmptyShortSequencePacketLE*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00,2004389 |

|  |  |
| --- | --- |
| **Llamada:**  public CDREncapsulation(IoBuffer buffer, object dataObj, ByteOrder order)  public static void Serialize(IoBuffer buffer, object dataObj, ByteOrder order)  public static T Deserialize<T>(IoBuffer buffer) | |
| ***Descripción*** | En esta prueba se muestra el corrector funcionamiento del CDREncapsulation para el EmptyEnumtSequencePacket Little Endian |
| ***Entrada*** | Dentro del Test Initialize se inicializa el serializador |
| ***Referencia*** |  |
| ***Código*** | [TestMethod]  public void TestEmptyEnumSequencePacketLE()  {  EnumSequencePacket v1 = new EnumSequencePacket(new MyEnum[] { });  int bufferSize = v1.Size + ArrayHeader + CDRHeaderSize;  var buffer = ByteBufferAllocator.Instance.Allocate(bufferSize);  CDREncapsulation.Serialize(buffer, v1, ByteOrder.LittleEndian);  Assert.AreEqual(bufferSize, buffer.Position);  buffer.Rewind();  Assert.AreEqual("00 01 00 00 00 00 00 00", buffer.GetHexDump());  EnumSequencePacket v2 = CDREncapsulation.Deserialize<EnumSequencePacket>(buffer);  Assert.AreEqual(v1, v2);  Assert.AreEqual(bufferSize, buffer.Position);  } |
| ***Salida*** | Nombre de la prueba: *TestEmptyEnumSequencePacketLE*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00,1785581 |

|  |  |
| --- | --- |
| **Llamada:**  public CDREncapsulation(IoBuffer buffer, object dataObj, ByteOrder order)  public static void Serialize(IoBuffer buffer, object dataObj, ByteOrder order)  public static T Deserialize<T>(IoBuffer buffer) | |
| ***Descripción*** | En esta prueba se muestra el corrector funcionamiento del CDREncapsulation para el EmptyEnumtSequencePacket Little Endian |
| ***Entrada*** | Dentro del Test Initialize se inicializa el serializador |
| ***Referencia*** |  |
| ***Código*** | [TestMethod]  public void TestEmptyIntSequencePacketLE()  {  IntSequencePacket v1 = new IntSequencePacket(new int[] { });  int bufferSize = v1.Size + ArrayHeader + CDRHeaderSize;  var buffer = ByteBufferAllocator.Instance.Allocate(bufferSize);  CDREncapsulation.Serialize(buffer, v1, ByteOrder.LittleEndian);  Assert.AreEqual(bufferSize, buffer.Position);  buffer.Rewind();  Assert.AreEqual("00 01 00 00 00 00 00 00", buffer.GetHexDump());  IntSequencePacket v2 = CDREncapsulation.Deserialize<IntSequencePacket>(buffer);  Assert.AreEqual(v1, v2);  Assert.AreEqual(bufferSize, buffer.Position);  } |
| ***Salida*** | Nombre de la prueba: *TestEmptyIntSequencePacketLE*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00,1852795 |

|  |  |
| --- | --- |
| **Llamada:**  public CDREncapsulation(IoBuffer buffer, object dataObj, ByteOrder order)  public static void Serialize(IoBuffer buffer, object dataObj, ByteOrder order)  public static T Deserialize<T>(IoBuffer buffer) | |
| ***Descripción*** | En esta prueba se muestra el corrector funcionamiento del CDREncapsulation para el BoolPacket Big Endian |
| ***Entrada*** | Dentro del Test Initialize se inicializa el serializador |
| ***Referencia*** |  |
| ***Código*** | [TestMethod]  public void TestBoolPacketBE()  {  BoolPacket v1 = new BoolPacket(true);  int bufferSize = sizeof(bool) + CDRHeaderSize;  var buffer = ByteBufferAllocator.Instance.Allocate(bufferSize);  CDREncapsulation.Serialize(buffer, v1, ByteOrder.BigEndian);  Assert.AreEqual(bufferSize, buffer.Position);  buffer.Rewind();  Assert.AreEqual("00 00 00 00 01", buffer.GetHexDump());  BoolPacket v2 = CDREncapsulation.Deserialize<BoolPacket>(buffer);  Assert.AreEqual(v1, v2);  Assert.AreEqual(bufferSize, buffer.Position);  } |
| ***Salida*** | Nombre de la prueba: *TestBoolPacketBE*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00,204271 |

|  |  |
| --- | --- |
| **Llamada:**  public CDREncapsulation(IoBuffer buffer, object dataObj, ByteOrder order)  public static void Serialize(IoBuffer buffer, object dataObj, ByteOrder order)  public static T Deserialize<T>(IoBuffer buffer) | |
| ***Descripción*** | En esta prueba se muestra el corrector funcionamiento del CDREncapsulation para el CharPacket Big Endian |
| ***Entrada*** | Dentro del Test Initialize se inicializa el serializador |
| ***Referencia*** |  |
| ***Código*** | [TestMethod]  public void TestCharPacketBE()  {  CharPacket v1 = new CharPacket('A');  int bufferSize = sizeof(char) + CDRHeaderSize;  var buffer = ByteBufferAllocator.Instance.Allocate(bufferSize);  CDREncapsulation.Serialize(buffer, v1, ByteOrder.BigEndian);  Assert.AreEqual(bufferSize, buffer.Position);  buffer.Rewind();  Assert.AreEqual("00 00 00 00 00 41", buffer.GetHexDump());  CharPacket v2 = CDREncapsulation.Deserialize<CharPacket>(buffer);  Assert.AreEqual(v1, v2);  Assert.AreEqual(bufferSize, buffer.Position);  } |
| ***Salida*** | Nombre de la prueba: *TestCharPacketBE*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00,1953506 |

|  |  |
| --- | --- |
| **Llamada:**  public CDREncapsulation(IoBuffer buffer, object dataObj, ByteOrder order)  public static void Serialize(IoBuffer buffer, object dataObj, ByteOrder order)  public static T Deserialize<T>(IoBuffer buffer) | |
| ***Descripción*** | En esta prueba se muestra el corrector funcionamiento del CDREncapsulation para el U8Packet Big Endian |
| ***Entrada*** | Dentro del Test Initialize se inicializa el serializador |
| ***Referencia*** |  |
| ***Código*** | [TestMethod]  public void TestU8PacketBE()  {  U8Packet v1 = new U8Packet(0xA);  int bufferSize = sizeof(byte) + CDRHeaderSize;  var buffer = ByteBufferAllocator.Instance.Allocate(bufferSize);  CDREncapsulation.Serialize(buffer, v1, ByteOrder.BigEndian);  Assert.AreEqual(bufferSize, buffer.Position);  buffer.Rewind();  Assert.AreEqual("00 00 00 00 0A", buffer.GetHexDump());  U8Packet v2 = CDREncapsulation.Deserialize<U8Packet>(buffer);  Assert.AreEqual(v1, v2);  Assert.AreEqual(bufferSize, buffer.Position);  } |
| ***Salida*** | Nombre de la prueba: *TestU8PackeBE*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00,2004217 |

|  |  |
| --- | --- |
| **Llamada:**  public CDREncapsulation(IoBuffer buffer, object dataObj, ByteOrder order)  public static void Serialize(IoBuffer buffer, object dataObj, ByteOrder order)  public static T Deserialize<T>(IoBuffer buffer) | |
| ***Descripción*** | En esta prueba se muestra el corrector funcionamiento del CDREncapsulation para el U16Packet Big Endian |
| ***Entrada*** | Dentro del Test Initialize se inicializa el serializador |
| ***Referencia*** |  |
| ***Código*** | [TestMethod]  public void TestU16PacketBE()  {  U16Packet v1 = new U16Packet(0xAB);  int bufferSize = sizeof(ushort) + CDRHeaderSize;  var buffer = ByteBufferAllocator.Instance.Allocate(bufferSize);  CDREncapsulation.Serialize(buffer, v1, ByteOrder.BigEndian);  Assert.AreEqual(bufferSize, buffer.Position);  buffer.Rewind();  Assert.AreEqual("00 00 00 00 00 AB", buffer.GetHexDump());  U16Packet v2 = CDREncapsulation.Deserialize<U16Packet>(buffer);  Assert.AreEqual(v1, v2);  Assert.AreEqual(bufferSize, buffer.Position);  } |
| ***Salida*** | Nombre de la prueba: *TestU16PacketBE*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00,2097027 |

|  |  |
| --- | --- |
| **Llamada:**  public CDREncapsulation(IoBuffer buffer, object dataObj, ByteOrder order)  public static void Serialize(IoBuffer buffer, object dataObj, ByteOrder order)  public static T Deserialize<T>(IoBuffer buffer) | |
| ***Descripción*** | En esta prueba se muestra el corrector funcionamiento del CDREncapsulation para el U32Packet Big Endian |
| ***Entrada*** | Dentro del Test Initialize se inicializa el serializador |
| ***Referencia*** |  |
| ***Código*** | [TestMethod]  public void TestU32PacketBE()  {  U32Packet v1 = new U32Packet(0xABA);  int bufferSize = sizeof(uint) + CDRHeaderSize;  var buffer = ByteBufferAllocator.Instance.Allocate(bufferSize);  CDREncapsulation.Serialize(buffer, v1, ByteOrder.BigEndian);  Assert.AreEqual(bufferSize, buffer.Position);  buffer.Rewind();  Assert.AreEqual("00 00 00 00 00 00 0A BA", buffer.GetHexDump());  U32Packet v2 = CDREncapsulation.Deserialize<U32Packet>(buffer);  Assert.AreEqual(v1, v2);  Assert.AreEqual(bufferSize, buffer.Position);  } |
| ***Salida*** | Nombre de la prueba: *TestU32PacketBE*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00,1931363 |

|  |  |
| --- | --- |
| **Llamada:**  public CDREncapsulation(IoBuffer buffer, object dataObj, ByteOrder order)  public static void Serialize(IoBuffer buffer, object dataObj, ByteOrder order)  public static T Deserialize<T>(IoBuffer buffer) | |
| ***Descripción*** | En esta prueba se muestra el corrector funcionamiento del CDREncapsulation para el U64Packet Big Endian |
| ***Entrada*** | Dentro del Test Initialize se inicializa el serializador |
| ***Referencia*** |  |
| ***Código*** | [TestMethod]  public void TestU64PacketBE()  {  U64Packet v1 = new U64Packet(0xABCDEF);  int bufferSize = sizeof(ulong) + CDRHeaderSize;  var buffer = ByteBufferAllocator.Instance.Allocate(bufferSize);  CDREncapsulation.Serialize(buffer, v1, ByteOrder.BigEndian);  Assert.AreEqual(bufferSize, buffer.Position);  buffer.Rewind();  Assert.AreEqual("00 00 00 00 00 00 00 00 00 AB CD EF", buffer.GetHexDump());  U64Packet v2 = CDREncapsulation.Deserialize<U64Packet>(buffer);  Assert.AreEqual(v1, v2);  Assert.AreEqual(bufferSize, buffer.Position);  } |
| ***Salida*** | Nombre de la prueba: *TestU64PacketBE*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00,2023983 |

|  |  |
| --- | --- |
| **Llamada:**  public CDREncapsulation(IoBuffer buffer, object dataObj, ByteOrder order)  public static void Serialize(IoBuffer buffer, object dataObj, ByteOrder order)  public static T Deserialize<T>(IoBuffer buffer) | |
| ***Descripción*** | En esta prueba se muestra el corrector funcionamiento del CDREncapsulation para el S8Packet Big Endian |
| ***Entrada*** | Dentro del Test Initialize se inicializa el serializador |
| ***Referencia*** |  |
| ***Código*** | [TestMethod]  public void TestS8PacketBE1()  {  S8Packet v1 = new S8Packet(-1);  int bufferSize = sizeof(sbyte) + CDRHeaderSize;  var buffer = ByteBufferAllocator.Instance.Allocate(bufferSize);  CDREncapsulation.Serialize(buffer, v1, ByteOrder.BigEndian);  Assert.AreEqual(bufferSize, buffer.Position);  buffer.Rewind();  Assert.AreEqual("00 00 00 00 FF", buffer.GetHexDump());  S8Packet v2 = CDREncapsulation.Deserialize<S8Packet>(buffer);  Assert.AreEqual(v1, v2);  Assert.AreEqual(bufferSize, buffer.Position);  } |
| ***Salida*** | Nombre de la prueba: *TestS8PacketBE1*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00,2144957 |

|  |  |
| --- | --- |
| **Llamada:**  public CDREncapsulation(IoBuffer buffer, object dataObj, ByteOrder order)  public static void Serialize(IoBuffer buffer, object dataObj, ByteOrder order)  public static T Deserialize<T>(IoBuffer buffer) | |
| ***Descripción*** | En esta prueba se muestra el corrector funcionamiento del CDREncapsulation para el S8Packet Big Endian |
| ***Entrada*** | Dentro del Test Initialize se inicializa el serializador |
| ***Referencia*** |  |
| ***Código*** | [TestMethod]  public void TestS8PacketBE2()  {  S8Packet v1 = new S8Packet(+1);  int bufferSize = sizeof(sbyte) + CDRHeaderSize;  var buffer = ByteBufferAllocator.Instance.Allocate(bufferSize);  CDREncapsulation.Serialize(buffer, v1, ByteOrder.BigEndian);  Assert.AreEqual(bufferSize, buffer.Position);  buffer.Rewind();  Assert.AreEqual("00 00 00 00 01", buffer.GetHexDump());  S8Packet v2 = CDREncapsulation.Deserialize<S8Packet>(buffer);  Assert.AreEqual(v1, v2);  Assert.AreEqual(bufferSize, buffer.Position);  } |
| ***Salida*** | Nombre de la prueba: *TestS8PacketBE2*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00,1883842 |

|  |  |
| --- | --- |
| **Llamada:**  public CDREncapsulation(IoBuffer buffer, object dataObj, ByteOrder order)  public static void Serialize(IoBuffer buffer, object dataObj, ByteOrder order)  public static T Deserialize<T>(IoBuffer buffer) | |
| ***Descripción*** | En esta prueba se muestra el corrector funcionamiento del CDREncapsulation para el S16Packet Big Endian |
| ***Entrada*** | Dentro del Test Initialize se inicializa el serializador |
| ***Referencia*** |  |
| ***Código*** | [TestMethod]  public void TestS16PacketBE1()  {  S16Packet v1 = new S16Packet(-10);  int bufferSize = sizeof(short) + CDRHeaderSize;  var buffer = ByteBufferAllocator.Instance.Allocate(bufferSize);  CDREncapsulation.Serialize(buffer, v1, ByteOrder.BigEndian);  Assert.AreEqual(bufferSize, buffer.Position);  buffer.Rewind();  Assert.AreEqual("00 00 00 00 FF F6", buffer.GetHexDump());  S16Packet v2 = CDREncapsulation.Deserialize<S16Packet>(buffer);  Assert.AreEqual(v1, v2);  Assert.AreEqual(bufferSize, buffer.Position);  } |
| ***Salida*** | Nombre de la prueba: *TestS16PacketBE1*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00,2190627 |

|  |  |
| --- | --- |
| **Llamada:**  public CDREncapsulation(IoBuffer buffer, object dataObj, ByteOrder order)  public static void Serialize(IoBuffer buffer, object dataObj, ByteOrder order)  public static T Deserialize<T>(IoBuffer buffer) | |
| ***Descripción*** | En esta prueba se muestra el corrector funcionamiento del CDREncapsulation para el S16Packet Big Endian |
| ***Entrada*** | Dentro del Test Initialize se inicializa el serializador |
| ***Referencia*** |  |
| ***Código*** | [TestMethod]  public void TestS16PacketBE2()  {  S16Packet v1 = new S16Packet(+10);  int bufferSize = sizeof(short) + CDRHeaderSize;  var buffer = ByteBufferAllocator.Instance.Allocate(bufferSize);  CDREncapsulation.Serialize(buffer, v1, ByteOrder.BigEndian);  Assert.AreEqual(bufferSize, buffer.Position);  buffer.Rewind();  Assert.AreEqual("00 00 00 00 00 0A", buffer.GetHexDump());  S16Packet v2 = CDREncapsulation.Deserialize<S16Packet>(buffer);  Assert.AreEqual(v1, v2);  Assert.AreEqual(bufferSize, buffer.Position);  } |
| ***Salida*** | Nombre de la prueba: *TestS16PacketBE2*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00,1791405 |

|  |  |
| --- | --- |
| **Llamada:**  public CDREncapsulation(IoBuffer buffer, object dataObj, ByteOrder order)  public static void Serialize(IoBuffer buffer, object dataObj, ByteOrder order)  public static T Deserialize<T>(IoBuffer buffer) | |
| ***Descripción*** | En esta prueba se muestra el corrector funcionamiento del CDREncapsulation para el S32Packet Big Endian |
| ***Entrada*** | Dentro del Test Initialize se inicializa el serializador |
| ***Referencia*** |  |
| ***Código*** | [TestMethod]  public void TestS32PacketBE1()  {  S32Packet v1 = new S32Packet(-0xABA);  int bufferSize = sizeof(int) + CDRHeaderSize;  var buffer = ByteBufferAllocator.Instance.Allocate(bufferSize);  CDREncapsulation.Serialize(buffer, v1, ByteOrder.BigEndian);  Assert.AreEqual(bufferSize, buffer.Position);  buffer.Rewind();  Assert.AreEqual("00 00 00 00 FF FF F5 46", buffer.GetHexDump());  S32Packet v2 = CDREncapsulation.Deserialize<S32Packet>(buffer);  Assert.AreEqual(v1, v2);  Assert.AreEqual(bufferSize, buffer.Position);  } |
| ***Salida*** | Nombre de la prueba: *TestS32PacketBE1*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00,2366255 |

|  |  |
| --- | --- |
| **Llamada:**  public CDREncapsulation(IoBuffer buffer, object dataObj, ByteOrder order)  public static void Serialize(IoBuffer buffer, object dataObj, ByteOrder order)  public static T Deserialize<T>(IoBuffer buffer) | |
| ***Descripción*** | En esta prueba se muestra el corrector funcionamiento del CDREncapsulation para el S32Packet Big Endian |
| ***Entrada*** | Dentro del Test Initialize se inicializa el serializador |
| ***Referencia*** |  |
| ***Código*** | [TestMethod]  public void TestS32PacketBE2()  {  S32Packet v1 = new S32Packet(0xABA);  int bufferSize = sizeof(int) + CDRHeaderSize;  var buffer = ByteBufferAllocator.Instance.Allocate(bufferSize);  CDREncapsulation.Serialize(buffer, v1, ByteOrder.BigEndian);  Assert.AreEqual(bufferSize, buffer.Position);  buffer.Rewind();  Assert.AreEqual("00 00 00 00 00 00 0A BA", buffer.GetHexDump());  S32Packet v2 = CDREncapsulation.Deserialize<S32Packet>(buffer);  Assert.AreEqual(v1, v2);  Assert.AreEqual(bufferSize, buffer.Position);  } |
| ***Salida*** | Nombre de la prueba: *TestS32PacketBE2*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00,1962171 |

|  |  |
| --- | --- |
| **Llamada:**  public CDREncapsulation(IoBuffer buffer, object dataObj, ByteOrder order)  public static void Serialize(IoBuffer buffer, object dataObj, ByteOrder order)  public static T Deserialize<T>(IoBuffer buffer) | |
| ***Descripción*** | En esta prueba se muestra el corrector funcionamiento del CDREncapsulation para el S4Packet Big Endian |
| ***Entrada*** | Dentro del Test Initialize se inicializa el serializador |
| ***Referencia*** |  |
| ***Código*** | [TestMethod]  public void TestS64PacketBE1()  {  S64Packet v1 = new S64Packet(-0xABCD);  int bufferSize = sizeof(long) + CDRHeaderSize;  var buffer = ByteBufferAllocator.Instance.Allocate(bufferSize);  CDREncapsulation.Serialize(buffer, v1, ByteOrder.BigEndian);  Assert.AreEqual(bufferSize, buffer.Position);  buffer.Rewind();  Assert.AreEqual("00 00 00 00 FF FF FF FF FF FF 54 33", buffer.GetHexDump());  S64Packet v2 = CDREncapsulation.Deserialize<S64Packet>(buffer);  Assert.AreEqual(v1, v2);  Assert.AreEqual(bufferSize, buffer.Position);  } |
| ***Salida*** | Nombre de la prueba: *TestS64PacketBE1*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00,2031737 |

|  |  |
| --- | --- |
| **Llamada:**  public CDREncapsulation(IoBuffer buffer, object dataObj, ByteOrder order)  public static void Serialize(IoBuffer buffer, object dataObj, ByteOrder order)  public static T Deserialize<T>(IoBuffer buffer) | |
| ***Descripción*** | En esta prueba se muestra el corrector funcionamiento del CDREncapsulation para el S4Packet Big Endian |
| ***Entrada*** | Dentro del Test Initialize se inicializa el serializador |
| ***Referencia*** |  |
| ***Código*** | [TestMethod]  public void TestS64PacketBE2()  {  S64Packet v1 = new S64Packet(0xABCD);  int bufferSize = sizeof(long) + CDRHeaderSize;  var buffer = ByteBufferAllocator.Instance.Allocate(bufferSize);  CDREncapsulation.Serialize(buffer, v1, ByteOrder.BigEndian);  Assert.AreEqual(bufferSize, buffer.Position);  buffer.Rewind();  Assert.AreEqual("00 00 00 00 00 00 00 00 00 00 AB CD", buffer.GetHexDump());  S64Packet v2 = CDREncapsulation.Deserialize<S64Packet>(buffer);  Assert.AreEqual(v1, v2);  Assert.AreEqual(bufferSize, buffer.Position);  } |
| ***Salida*** | Nombre de la prueba: *TestS64PacketLE2*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00,1994617 |

|  |  |
| --- | --- |
| **Llamada:**  public CDREncapsulation(IoBuffer buffer, object dataObj, ByteOrder order)  public static void Serialize(IoBuffer buffer, object dataObj, ByteOrder order)  public static T Deserialize<T>(IoBuffer buffer) | |
| ***Descripción*** | En esta prueba se muestra el corrector funcionamiento del CDREncapsulation para el SinglePacket Big Endian |
| ***Entrada*** | Dentro del Test Initialize se inicializa el serializador |
| ***Referencia*** |  |
| ***Código*** | [TestMethod]  public void TestSinglePacketBE()  {  SinglePacket v1 = new SinglePacket(0.1f);  int bufferSize = sizeof(float) + CDRHeaderSize;  var buffer = ByteBufferAllocator.Instance.Allocate(bufferSize);  CDREncapsulation.Serialize(buffer, v1, ByteOrder.BigEndian);  Assert.AreEqual(bufferSize, buffer.Position);  buffer.Rewind();  Assert.AreEqual("00 00 00 00 3D CC CC CD", buffer.GetHexDump());  SinglePacket v2 = CDREncapsulation.Deserialize<SinglePacket>(buffer);  Assert.AreEqual(v1, v2);  Assert.AreEqual(bufferSize, buffer.Position);  } |
| ***Salida*** | Nombre de la prueba: *TestSinglePacketBE*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00,1785456 |

|  |  |
| --- | --- |
| **Llamada:**  public CDREncapsulation(IoBuffer buffer, object dataObj, ByteOrder order)  public static void Serialize(IoBuffer buffer, object dataObj, ByteOrder order)  public static T Deserialize<T>(IoBuffer buffer) | |
| ***Descripción*** | En esta prueba se muestra el corrector funcionamiento del CDREncapsulation para el DoublePacket Big Endian |
| ***Entrada*** | Dentro del Test Initialize se inicializa el serializador |
| ***Referencia*** |  |
| ***Código*** | [TestMethod]  public void TestDoublePacketBE()  {  DoublePacket v1 = new DoublePacket(0.1);  int bufferSize = sizeof(double) + CDRHeaderSize;  var buffer = ByteBufferAllocator.Instance.Allocate(bufferSize);  CDREncapsulation.Serialize(buffer, v1, ByteOrder.BigEndian);  Assert.AreEqual(bufferSize, buffer.Position);  buffer.Rewind();  Assert.AreEqual("00 00 00 00 3F B9 99 99 99 99 99 9A", buffer.GetHexDump());  DoublePacket v2 = CDREncapsulation.Deserialize<DoublePacket>(buffer);  Assert.AreEqual(v1, v2);  Assert.AreEqual(bufferSize, buffer.Position);  } |
| ***Salida*** | Nombre de la prueba: *TestDoublePacketLE*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00,199862 |

|  |  |
| --- | --- |
| **Llamada:**  public CDREncapsulation(IoBuffer buffer, object dataObj, ByteOrder order)  public static void Serialize(IoBuffer buffer, object dataObj, ByteOrder order)  public static T Deserialize<T>(IoBuffer buffer) | |
| ***Descripción*** | En esta prueba se muestra el corrector funcionamiento del CDREncapsulation para el BoolSequencePacket Big Endian |
| ***Entrada*** | Dentro del Test Initialize se inicializa el serializador |
| ***Referencia*** |  |
| ***Código*** | [TestMethod]  public void TestBoolSequencePacketBE()  {  BoolSequencePacket v1 = new BoolSequencePacket(new bool[] { true, false, false, true });  int bufferSize = v1.Size + ArrayHeader + CDRHeaderSize;  var buffer = ByteBufferAllocator.Instance.Allocate(bufferSize);  CDREncapsulation.Serialize(buffer, v1, ByteOrder.BigEndian);  Assert.AreEqual(bufferSize, buffer.Position);  buffer.Rewind();  Assert.AreEqual("00 00 00 00 00 00 00 04 01 00 00 01", buffer.GetHexDump());  BoolSequencePacket v2 = CDREncapsulation.Deserialize<BoolSequencePacket>(buffer);  Assert.AreEqual(v1, v2);  Assert.AreEqual(bufferSize, buffer.Position);  } |
| ***Salida*** | Nombre de la prueba: *TestBoolSequencePacketBE*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00,1890411 |

|  |  |
| --- | --- |
| **Llamada:**  public CDREncapsulation(IoBuffer buffer, object dataObj, ByteOrder order)  public static void Serialize(IoBuffer buffer, object dataObj, ByteOrder order)  public static T Deserialize<T>(IoBuffer buffer) | |
| ***Descripción*** | En esta prueba se muestra el corrector funcionamiento del CDREncapsulation para el ShortSequencePacket Big Endian |
| ***Entrada*** | Dentro del Test Initialize se inicializa el serializador |
| ***Referencia*** |  |
| ***Código*** | [TestMethod]  public void TestShortSequencePacketBE()  {  ShortSequencePacket v1 = new ShortSequencePacket(new short[] { 0xFA1, 0xFF0, 0xB2F, 0x001 });  int bufferSize = v1.Size + ArrayHeader + CDRHeaderSize;  var buffer = ByteBufferAllocator.Instance.Allocate(bufferSize);  CDREncapsulation.Serialize(buffer, v1, ByteOrder.BigEndian);  Assert.AreEqual(bufferSize, buffer.Position);  buffer.Rewind();  Assert.AreEqual("00 00 00 00 00 00 00 04 0F A1 0F F0 0B 2F 00 01", buffer.GetHexDump());  ShortSequencePacket v2 = CDREncapsulation.Deserialize<ShortSequencePacket>(buffer);  Assert.AreEqual(v1, v2);  Assert.AreEqual(bufferSize, buffer.Position);  } |
| ***Salida*** | Nombre de la prueba: *TestShortSequencePacketBE*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00,1981403 |

|  |  |
| --- | --- |
| **Llamada:**  public CDREncapsulation(IoBuffer buffer, object dataObj, ByteOrder order)  public static void Serialize(IoBuffer buffer, object dataObj, ByteOrder order)  public static T Deserialize<T>(IoBuffer buffer) | |
| ***Descripción*** | En esta prueba se muestra el corrector funcionamiento del CDREncapsulation para el EnumSequencePacket Big Endian |
| ***Entrada*** | Dentro del Test Initialize se inicializa el serializador |
| ***Referencia*** |  |
| ***Código*** | [TestMethod]  public void TestEnumSequencePacketBE()  {  EnumSequencePacket v1 = new EnumSequencePacket(new MyEnum[] { MyEnum.Four, MyEnum.Three, MyEnum.Three, MyEnum.Zero, MyEnum.One, MyEnum.Five });  int bufferSize = v1.Size + ArrayHeader + CDRHeaderSize;  var buffer = ByteBufferAllocator.Instance.Allocate(bufferSize);  CDREncapsulation.Serialize(buffer, v1, ByteOrder.BigEndian);  Assert.AreEqual(bufferSize, buffer.Position);  buffer.Rewind();  Assert.AreEqual("00 00 00 00 00 00 00 06 00 00 00 04 00 00 00 03 00 00 00 03 00 00 00 00 00 00 00 01 00 00 00 05", buffer.GetHexDump());  EnumSequencePacket v2 = CDREncapsulation.Deserialize<EnumSequencePacket>(buffer);  Assert.AreEqual(v1, v2);  Assert.AreEqual(bufferSize, buffer.Position);  } |
| ***Salida*** | Nombre de la prueba: *TestEnumSequencePacketBE*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00,1841432 |

|  |  |
| --- | --- |
| **Llamada:**  public CDREncapsulation(IoBuffer buffer, object dataObj, ByteOrder order)  public static void Serialize(IoBuffer buffer, object dataObj, ByteOrder order)  public static T Deserialize<T>(IoBuffer buffer) | |
| ***Descripción*** | En esta prueba se muestra el corrector funcionamiento del CDREncapsulation para el IntSequencePacket Big Endian |
| ***Entrada*** | Dentro del Test Initialize se inicializa el serializador |
| ***Referencia*** |  |
| ***Código*** | [TestMethod]  public void TestIntSequencePacketBE()  {  IntSequencePacket v1 = new IntSequencePacket(new int[] { 0xFFA1F0, 0xFF230F, 0xB2000F, 0xFFFFF01 });  int bufferSize = v1.Size + ArrayHeader + CDRHeaderSize;  var buffer = ByteBufferAllocator.Instance.Allocate(bufferSize);  CDREncapsulation.Serialize(buffer, v1, ByteOrder.BigEndian);  Assert.AreEqual(bufferSize, buffer.Position);  buffer.Rewind();  Assert.AreEqual("00 00 00 00 00 00 00 04 00 FF A1 F0 00 FF 23 0F 00 B2 00 0F 0F FF FF 01", buffer.GetHexDump());  IntSequencePacket v2 = CDREncapsulation.Deserialize<IntSequencePacket>(buffer);  Assert.AreEqual(v1, v2);  Assert.AreEqual(bufferSize, buffer.Position);  } |
| ***Salida*** | Nombre de la prueba: *TestIntSequencePacketBE*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00,1821422 |

|  |  |
| --- | --- |
| **Llamada:**  public CDREncapsulation(IoBuffer buffer, object dataObj, ByteOrder order)  public static void Serialize(IoBuffer buffer, object dataObj, ByteOrder order)  public static T Deserialize<T>(IoBuffer buffer) | |
| ***Descripción*** | En esta prueba se muestra el corrector funcionamiento del CDREncapsulation para el EmptyBoolSequencePacket Big Endian |
| ***Entrada*** | Dentro del Test Initialize se inicializa el serializador |
| ***Referencia*** |  |
| ***Código*** | [TestMethod]  public void TestEmptyBoolSequencePacketBE()  {  BoolSequencePacket v1 = new BoolSequencePacket(new bool[] { });  int bufferSize = v1.Size + ArrayHeader + CDRHeaderSize;  var buffer = ByteBufferAllocator.Instance.Allocate(bufferSize);  CDREncapsulation.Serialize(buffer, v1, ByteOrder.BigEndian);  Assert.AreEqual(bufferSize, buffer.Position);  buffer.Rewind();  Assert.AreEqual("00 00 00 00 00 00 00 00", buffer.GetHexDump());  BoolSequencePacket v2 = CDREncapsulation.Deserialize<BoolSequencePacket>(buffer);  Assert.AreEqual(v1, v2);  Assert.AreEqual(bufferSize, buffer.Position);  } |
| ***Salida*** | Nombre de la prueba: *TestEmptyBoolSequencePacketBE*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00,1865145 |

|  |  |
| --- | --- |
| **Llamada:**  public CDREncapsulation(IoBuffer buffer, object dataObj, ByteOrder order)  public static void Serialize(IoBuffer buffer, object dataObj, ByteOrder order)  public static T Deserialize<T>(IoBuffer buffer) | |
| ***Descripción*** | En esta prueba se muestra el corrector funcionamiento del CDREncapsulation para el EmptyShortSequencePacket Big Endian |
| ***Entrada*** | Dentro del Test Initialize se inicializa el serializador |
| ***Referencia*** |  |
| ***Código*** | [TestMethod]  public void TestEmptyShortSequencePacketBE()  {  ShortSequencePacket v1 = new ShortSequencePacket(new short[] { });  int bufferSize = v1.Size + ArrayHeader + CDRHeaderSize;  var buffer = ByteBufferAllocator.Instance.Allocate(bufferSize);  CDREncapsulation.Serialize(buffer, v1, ByteOrder.BigEndian);  Assert.AreEqual(bufferSize, buffer.Position);  buffer.Rewind();  Assert.AreEqual("00 00 00 00 00 00 00 00", buffer.GetHexDump());  ShortSequencePacket v2 = CDREncapsulation.Deserialize<ShortSequencePacket>(buffer);  Assert.AreEqual(v1, v2);  Assert.AreEqual(bufferSize, buffer.Position);  } |
| ***Salida*** | Nombre de la prueba: *TestEmptyShortSequencePacketBE*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00,2001416 |

|  |  |
| --- | --- |
| **Llamada:**  public CDREncapsulation(IoBuffer buffer, object dataObj, ByteOrder order)  public static void Serialize(IoBuffer buffer, object dataObj, ByteOrder order)  public static T Deserialize<T>(IoBuffer buffer) | |
| ***Descripción*** | En esta prueba se muestra el corrector funcionamiento del CDREncapsulation para el EmptyEnumtSequencePacket Big Endian |
| ***Entrada*** | Dentro del Test Initialize se inicializa el serializador |
| ***Referencia*** |  |
| ***Código*** | [TestMethod]  public void TestEmptyEnumSequencePacketBE()  {  EnumSequencePacket v1 = new EnumSequencePacket(new MyEnum[] { });  int bufferSize = v1.Size + ArrayHeader + CDRHeaderSize;  var buffer = ByteBufferAllocator.Instance.Allocate(bufferSize);  CDREncapsulation.Serialize(buffer, v1, ByteOrder.BigEndian);  Assert.AreEqual(bufferSize, buffer.Position);  buffer.Rewind();  Assert.AreEqual("00 00 00 00 00 00 00 00", buffer.GetHexDump());  EnumSequencePacket v2 = CDREncapsulation.Deserialize<EnumSequencePacket>(buffer);  Assert.AreEqual(v1, v2);  Assert.AreEqual(bufferSize, buffer.Position);  } |
| ***Salida*** | Nombre de la prueba: *TestEmptyEnumSequencePacketBE*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00,1854031 |

|  |  |
| --- | --- |
| **Llamada:**  public CDREncapsulation(IoBuffer buffer, object dataObj, ByteOrder order)  public static void Serialize(IoBuffer buffer, object dataObj, ByteOrder order)  public static T Deserialize<T>(IoBuffer buffer) | |
| ***Descripción*** | En esta prueba se muestra el corrector funcionamiento del CDREncapsulation para el EmptyEnumtSequencePacket Big Endian |
| ***Entrada*** | Dentro del Test Initialize se inicializa el serializador |
| ***Referencia*** |  |
| ***Código*** | [TestMethod]  public void TestEmptyIntSequencePacketBE()  {  IntSequencePacket v1 = new IntSequencePacket(new int[] { });  int bufferSize = v1.Size + ArrayHeader + CDRHeaderSize;  var buffer = ByteBufferAllocator.Instance.Allocate(bufferSize);  CDREncapsulation.Serialize(buffer, v1, ByteOrder.BigEndian);  Assert.AreEqual(bufferSize, buffer.Position);  buffer.Rewind();  Assert.AreEqual("00 00 00 00 00 00 00 00", buffer.GetHexDump());  IntSequencePacket v2 = CDREncapsulation.Deserialize<IntSequencePacket>(buffer);  Assert.AreEqual(v1, v2);  Assert.AreEqual(bufferSize, buffer.Position);  } |
| ***Salida*** | Nombre de la prueba: *TestEmptyIntSequencePacketLE*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00,1957202 |

#### Pruebas de exploración de tipo.

|  |  |
| --- | --- |
| **Llamada:**  public static org.omg.dds.type.typeobject.Type ExploreType(System.Type type) | |
| ***Descripción*** | En esta prueba se verifica el correcto funcionamiento del serializador de DDS en una Clase |
| ***Entrada*** | Inicialmente no se tiene inicializado al ddsType |
| ***Referencia*** |  |
| ***Código*** | [TestMethod]  public void TestExploreMyClass1()  {  var ddsType = TypeExplorer.ExploreType(typeof(XMyClass1));  Assert.IsNotNull(ddsType);  Assert.IsNotNull(ddsType.GetProperty());  var propInfo = ddsType.GetProperty();  Assert.AreEqual("SerializerTests.XMyClass1", propInfo.Name);  Assert.IsInstanceOfType(ddsType, typeof(StructureType));  StructureType structType = ddsType as StructureType;  var members = structType.GetMember();  Assert.IsNotNull(members);  Assert.AreEqual(3, members.Count);  Assert.AreEqual("m\_byte", members[0].GetProperty().Name);  Assert.AreEqual("m\_int", members[1].GetProperty().Name);  Assert.AreEqual("m\_short", members[2].GetProperty().Name);  Assert.AreEqual((uint)0x8001, members[0].GetProperty().MemberId);  Assert.AreEqual((uint)0x8002, members[1].GetProperty().MemberId);  Assert.AreEqual((uint)0x8003, members[2].GetProperty().MemberId); |
| ***Salida*** | Nombre de la prueba: *TestExploreMyClass1*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00,1996965 |

#### Pruebas de paquetes.

#### Pruebas de primitivas.

|  |  |
| --- | --- |
| **Llamada:**  App.config | |
| ***Descripción*** | En esta prueba se verifica que el fichero de configuración no sea nulo |
| ***Entrada*** | Inicialmente se tiene el fichero de configuración |
| ***Referencia*** |  |
| ***Código*** | public void TestExistConfiguration()  {  Assert.IsNotNull(ddsConfig);  } |
| ***Salida*** | Nombre de la prueba: TestExistConfiguration  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00,0286387 |