# DOCUMENTO DE PRUEBAS UNITARIAS DEL API RTPS.

# INTRODUCCIÓN

Paquetes del API a probar:

**Codificadores:** Contiene todas las clases referentes a la codificación, con los métodos respectivos para codificar diferentes tipos de información.

**Transporte:** Dentro de esta carpeta se encuentra el corazón de RTPS, ya que tiene las funcionalidades principales del API RTPS, como por ejemplo a los diferentes tipos de mensaje, a los escritores y lectores con y sin estado.

**Utilidades:** En esta carpeta se puede encontrar código que es útil al momento de envió de mensajes constante o un generador de identidad automático.

**Serializador:** El serializador no es parte de las carpetas de RTPS pero es esencial que este sea probado ya que se encuentran todos los tipos de paquetes, tipos de dato, se encarga de la serialización de datos y trabaja junto al API de DDS, se puede decir que es parte de la conexión de DDS con RTPS.

.

# PRUEBAS

En esta sección se detallan las pruebas que se van realizando en cada uno de los componentes del API RTPS y el Serializador.

## PRUEBAS UNITARIAS DEL API RTPS

### Codificadores

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

Tabla ‑. TestLocatorIpV4CDR\_BE

|  |  |
| --- | --- |
| **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 los *Locator* con el *Serializador* y el *Deserealizador.* |
| ***Entrada*** | Inicialmente no existe ninguna instancia del *DataEncapsulation.* |
| ***Referencia*** | **9.4.3.1** |
| ***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: *TestLocatorIpV4CDR\_BE*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:02.9120332 |

Tabla ‑. TestLocatorIpV4CDR\_LE

|  |  |
| --- | --- |
| **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 los *Locator* con el *Serializador* y el *Deserealizador.* |
| ***Entrada*** | Inicialmente no existe ninguna instancia del *DataEncapsulation.* |
| ***Referencia*** | 9.4.3.1 |
| ***Código*** | [TestMethod]  public void TestLocatorIpV4CDR\_LE()  {  Encapsulation Scheme = Encapsulation.CDR\_LE;  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 01 00 00 01 00 00 00 8C 0A 00 00 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: *TestLocatorIpV4CDR\_LE*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 00:00:00.1384335 |

Tabla ‑. TestLocatorIpV6CDR\_BE

|  |  |
| --- | --- |
| **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 los *Locator* con el *Serializador* y el *Deserealizador.* |
| ***Entrada*** | Inicialmente no existe ninguna instancia del *DataEncapsulation.* |
| ***Referencia*** | 9.4.3.1 |
| ***Código*** | [TestMethod]  public void TestLocatorIpV6CDR\_BE()  {  Encapsulation Scheme = Encapsulation.CDR\_BE;  int bufferSize = 16 + 4 + 4 + CDRHeaderSize;  Locator v1 = new Locator(IPAddress.Parse("::1"), 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 02 00 00 0A 8C 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01", buffer.GetHexDump());  Locator v2 = EncapsulationManager.Deserialize<Locator>(buffer);  Assert.AreEqual(v1, v2);  } |
| ***Salida*** | Nombre de la prueba: *TestLocatorIpV6CDR\_BE*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00.1594283 |

*Tabla 4‑4. TestLocatorIpV6CDR\_LE*

|  |  |
| --- | --- |
| **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 los *Locator* con el *Serializador* y el *Deserealizador.* |
| ***Entrada*** | Inicialmente no existe ninguna instancia del *DataEncapsulation.* |
| ***Referencia*** | 9.4.3.1 |
| ***Código*** | [TestMethod]  public void TestLocatorIpV6CDR\_LE()  {  Encapsulation Scheme = Encapsulation.CDR\_LE;  int bufferSize = 16 + 4 + 4 + CDRHeaderSize;  Locator v1 = new Locator(IPAddress.Parse("::1"), 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 01 00 00 02 00 00 00 8C 0A 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01", buffer.GetHexDump());  Locator v2 = EncapsulationManager.Deserialize<Locator>(buffer);  Assert.AreEqual(v1, v2);  } |
| ***Salida*** | Nombre de la prueba: *TestLocatorIpV6CDR\_LE*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00.1602378 |

Tabla ‑. TestLocatorIpV6CDR\_BE2

|  |  |
| --- | --- |
| **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 los *Locator* con el *Serializador* y el *Deserealizador.* |
| ***Entrada*** | Inicialmente no existe ninguna instancia del *DataEncapsulation.* |
| ***Referencia*** | 9.4.3.1 |
| ***Código*** | [TestMethod]  public void TestLocatorIpV6CDR\_BE2()  {  Encapsulation Scheme = Encapsulation.CDR\_BE;  int bufferSize = 16 + 4 + 4 + CDRHeaderSize;  Locator v1 = new Locator(IPAddress.Parse("FF00:4501:0:0:0:0:0:32"), 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 02 00 00 0A 8C FF 00 45 01 00 00 00 00 00 00 00 00 00 00 00 32", buffer.GetHexDump());  Locator v2 = EncapsulationManager.Deserialize<Locator>(buffer);  Assert.AreEqual(v1, v2);  } |
| ***Salida*** | Nombre de la prueba: *TestLocatorIpV6CDR\_BE2*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00.1492697 |

Tabla ‑. TestLocatorIpV6CDR\_LE2

|  |  |
| --- | --- |
| **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 los *Locator* con el *Serializador* y el *Deserealizador.* |
| ***Entrada*** | Inicialmente no existe ninguna instancia del *DataEncapsulation.* |
| ***Referencia*** | 9.4.3.1 |
| ***Código*** | [TestMethod]  public void TestLocatorIpV6CDR\_LE2()  {  Encapsulation Scheme = Encapsulation.CDR\_LE;  int bufferSize = 16 + 4 + 4 + CDRHeaderSize;  Locator v1 = new Locator(IPAddress.Parse("FF00:4501:0:0:0:0:0:32"), 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 01 00 00 02 00 00 00 8C 0A 00 00 FF 00 45 01 00 00 00 00 00 00 00 00 00 00 00 32", buffer.GetHexDump());  Locator v2 = EncapsulationManager.Deserialize<Locator>(buffer);  Assert.AreEqual(v1, v2);  } |
| ***Salida*** | Nombre de la prueba: *TestLocatorIpV6CDR\_LE2*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00.1643104 |

Tabla ‑. TestLocatorFromSample1

|  |  |
| --- | --- |
| **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 el *Locator* con el *Serializador* y el *Deserealizador.* |
| ***Entrada*** | Inicialmente no existe ninguna instancia del *DataEncapsulation.* |
| ***Referencia*** | 10.2.1 |
| ***Código*** | [TestMethod]  public void TestLocatorFromSample1()  {  Encapsulation Scheme = Encapsulation.CDR\_LE;  int bufferSize = 16 + 4 + 4 + CDRHeaderSize;  Locator v1 = new Locator(IPAddress.Parse("172.16.0.128"), 36945);  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 01 00 00 01 00 00 00 51 90 00 00 00 00 00 00 00 00 00 00 00 00 00 00 AC 10 00 80", buffer.GetHexDump());  Locator v2 = EncapsulationManager.Deserialize<Locator>(buffer);  Assert.AreEqual(v1, v2);  } |
| ***Salida*** | Nombre de la prueba: *TestLocatorFromSample1*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00.172042 |

*Tabla 4‑8. TestLocatorFromSample2*

|  |  |
| --- | --- |
| **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 el *Locator* con el *Serializador* y el *Deserealizador.* |
| ***Entrada*** | Inicialmente no existe ninguna instancia del *DataEncapsulation.* |
| ***Referencia*** | 10.2.1 |
| ***Código*** | [TestMethod]  public void TestLocatorFromSample2()  {  Encapsulation Scheme = Encapsulation.CDR\_LE;  int bufferSize = 16 + 4 + 4 + CDRHeaderSize;  Locator v1 = new Locator(IPAddress.Parse("239.255.0.1"), 9652);  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 01 00 00 01 00 00 00 B4 25 00 00 00 00 00 00 00 00 00 00 00 00 00 00 EF FF 00 01", buffer.GetHexDump());  Locator v2 = EncapsulationManager.Deserialize<Locator>(buffer);  Assert.AreEqual(v1, v2);  } |
| ***Salida*** | Nombre de la prueba: *TestLocatorFromSample2*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00.1519815 |

Tabla ‑. TestLocatorFromSample3

|  |  |
| --- | --- |
| **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 el *Locator* con el *Serializador* y el *Deserealizador.* |
| ***Entrada*** | Inicialmente no existe ninguna instancia del *DataEncapsulation.* |
| ***Referencia*** | 10.2.1 |
| ***Código*** | [TestMethod]  public void TestLocatorFromSample3()  {  Encapsulation Scheme = Encapsulation.CDR\_LE;  int bufferSize = 16 + 4 + 4 + CDRHeaderSize;  Locator v1 = new Locator(IPAddress.Parse("127.0.0.1"), 12345);  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 01 00 00 01 00 00 00 39 30 00 00 00 00 00 00 00 00 00 00 00 00 00 00 7F 00 00 01", buffer.GetHexDump());  Locator v2 = EncapsulationManager.Deserialize<Locator>(buffer);  Assert.AreEqual(v1, v2);  } |
| ***Salida*** | Nombre de la prueba: *TestLocatorFromSample3*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00.1650963 |

Tabla ‑. TestLocatorIpV4PL\_CDR\_BE

|  |  |
| --- | --- |
| **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 el *ClassWithLocator* con el *Serializador* y el *Deserealizador.* |
| ***Entrada*** | Inicialmente no existe ninguna instancia del *DataEncapsulation.* |
| ***Referencia*** | 10.2.1 |
| ***Código*** | [TestMethod]  public void TestLocatorIpV4PL\_CDR\_\_BE()  {  Encapsulation Scheme = Encapsulation.PL\_CDR\_BE;  int bufferSize = 16 + 4 + 4 + PL\_CDRHeaderSize;  ClassWithLocator v1 = new ClassWithLocator() { Locator = new Locator(IPAddress.Parse("10.20.30.40"), 2700) };  SerializedPayload payload = new SerializedPayload();  payload.DataEncapsulation = EncapsulationManager.Serialize<ClassWithLocator>(v1, Scheme);  IoBuffer buffer = IoBuffer.Wrap(payload.DataEncapsulation.SerializedPayload);  Assert.AreEqual(bufferSize, buffer.Remaining);  Assert.AreEqual("00 02 00 00 00 48 00 18 00 00 00 01 00 00 0A 8C 00 00 00 00 00 00 00 00 00 00 00 00 0A 14 1E 28 00 01 00 00", buffer.GetHexDump());  ClassWithLocator v2 = EncapsulationManager.Deserialize<ClassWithLocator>(buffer);  Assert.AreEqual(v1.Locator, v2.Locator);  } |
| ***Salida*** | Nombre de la prueba: *TestLocatorIpV4PL\_CDR\_BE*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00.1744933 |

Tabla ‑. TestLocatorIpV4PL\_CDR\_LE

|  |  |
| --- | --- |
| **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 el *ClassWithLocator* con el *Serializador* y el *Deserealizador.* |
| ***Entrada*** | Inicialmente no existe ninguna instancia del *DataEncapsulation.* |
| ***Referencia*** | 10.2.1 |
| ***Código*** | [TestMethod]  public void TestLocatorIpV4PL\_CDR\_LE()  {  Encapsulation Scheme = Encapsulation.PL\_CDR\_LE;  int bufferSize = 16 + 4 + 4 + PL\_CDRHeaderSize;  ClassWithLocator v1 = new ClassWithLocator() { Locator = new Locator(IPAddress.Parse("10.20.30.40"), 2700) };  SerializedPayload payload = new SerializedPayload();  payload.DataEncapsulation = EncapsulationManager.Serialize<ClassWithLocator>(v1, Scheme);  IoBuffer buffer = IoBuffer.Wrap(payload.DataEncapsulation.SerializedPayload);  Assert.AreEqual(bufferSize, buffer.Remaining);  Assert.AreEqual("00 03 00 00 48 00 18 00 01 00 00 00 8C 0A 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0A 14 1E 28 01 00 00 00", buffer.GetHexDump());  ClassWithLocator v2 = EncapsulationManager.Deserialize<ClassWithLocator>(buffer);  Assert.AreEqual(v1.Locator, v2.Locator);  } |
| ***Salida*** | Nombre de la prueba: *TestLocatorIpV4PL\_CDR\_LE*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00.1634546 |

Tabla ‑. TestLocatorIpV6PL\_CDR\_BE

|  |  |
| --- | --- |
| **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 el *ClassWithLocator* con el *Serializador* y el *Deserealizador.* |
| ***Entrada*** | Inicialmente no existe ninguna instancia del *DataEncapsulation.* |
| ***Referencia*** | 9.4.3.1 |
| ***Código*** | [TestMethod]  public void TestLocatorIpV6PL\_CDR\_BE()  {  Encapsulation Scheme = Encapsulation.PL\_CDR\_BE;  int bufferSize = 16 + 4 + 4 + PL\_CDRHeaderSize;  ClassWithLocator v1 = new ClassWithLocator() { Locator = new Locator(IPAddress.Parse("::1"), 2700) };  SerializedPayload payload = new SerializedPayload();  payload.DataEncapsulation = EncapsulationManager.Serialize<ClassWithLocator>(v1, Scheme);  IoBuffer buffer = IoBuffer.Wrap(payload.DataEncapsulation.SerializedPayload);  Assert.AreEqual(bufferSize, buffer.Remaining);  Assert.AreEqual("00 02 00 00 00 48 00 18 00 00 00 02 00 00 0A 8C 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 00 01 00 00", buffer.GetHexDump());  ClassWithLocator v2 = EncapsulationManager.Deserialize<ClassWithLocator>(buffer);  Assert.AreEqual(v1.Locator, v2.Locator);  } |
| ***Salida*** | Nombre de la prueba: *TestLocatorIpV6PL\_CDR\_BE*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00.1639482 |

Tabla ‑. TestLocatorIpV6PL\_CDR\_LE

|  |  |
| --- | --- |
| **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 el *ClassWithLocator* con el *Serializador* y el *Deserealizador.* |
| ***Entrada*** | Inicialmente no existe ninguna instancia del *DataEncapsulation.* |
| ***Referencia*** | 9.4.3.1 |
| ***Código*** | [TestMethod]  public void TestLocatorIpV6PL\_CDR\_LE()  {  Encapsulation Scheme = Encapsulation.PL\_CDR\_LE;  int bufferSize = 16 + 4 + 4 + PL\_CDRHeaderSize;  ClassWithLocator v1 = new ClassWithLocator() { Locator = new Locator(IPAddress.Parse("::1"), 2700) };  SerializedPayload payload = new SerializedPayload();  payload.DataEncapsulation = EncapsulationManager.Serialize<ClassWithLocator>(v1, Scheme);  IoBuffer buffer = IoBuffer.Wrap(payload.DataEncapsulation.SerializedPayload);  Assert.AreEqual(bufferSize, buffer.Remaining);  Assert.AreEqual("00 03 00 00 48 00 18 00 02 00 00 00 8C 0A 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 01 00 00 00", buffer.GetHexDump());  ClassWithLocator v2 = EncapsulationManager.Deserialize<ClassWithLocator>(buffer);  Assert.AreEqual(v1.Locator, v2.Locator);  } |
| ***Salida*** | Nombre de la prueba: *TestLocatorIpV6PL\_CDR\_LE*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00.1558723 |

Tabla ‑. TestLocatorIpV6PL\_CDR\_BE2

|  |  |
| --- | --- |
| **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 el *ClassWithLocator* con el *Serializador* y el *Deserealizador.* |
| ***Entrada*** | Inicialmente no existe ninguna instancia del *DataEncapsulation.* |
| ***Referencia*** | 9.4.3.1 |
| ***Código*** | [TestMethod]  public void TestLocatorIpV6PL\_CDR\_BE2()  {  Encapsulation Scheme = Encapsulation.PL\_CDR\_BE;  int bufferSize = 16 + 4 + 4 + PL\_CDRHeaderSize;  ClassWithLocator v1 = new ClassWithLocator() { Locator = new Locator(IPAddress.Parse("FF00:4501:0:0:0:0:0:32"), 2700) };  SerializedPayload payload = new SerializedPayload();  payload.DataEncapsulation = EncapsulationManager.Serialize<ClassWithLocator>(v1, Scheme);  IoBuffer buffer = IoBuffer.Wrap(payload.DataEncapsulation.SerializedPayload);  Assert.AreEqual(bufferSize, buffer.Remaining);  Assert.AreEqual("00 02 00 00 00 48 00 18 00 00 00 02 00 00 0A 8C FF 00 45 01 00 00 00 00 00 00 00 00 00 00 00 32 00 01 00 00", buffer.GetHexDump());  ClassWithLocator v2 = EncapsulationManager.Deserialize<ClassWithLocator>(buffer);  Assert.AreEqual(v1.Locator, v2.Locator);  } |
| ***Salida*** | Nombre de la prueba: *TestLocatorIpV6PL\_CDR\_BE2*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 00:00:00.1640392 |

Tabla ‑. TestLocatorIpV6PL\_CDR\_LE2

|  |  |
| --- | --- |
| **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 el *ClassWithLocator* con el *Serializador* y el *Deserealizador.* |
| ***Entrada*** | Inicialmente no existe ninguna instancia del *DataEncapsulation.* |
| ***Referencia*** | 9.4.3.1 |
| ***Código*** | [TestMethod]  public void TestLocatorIpV6PL\_CDR\_LE2()  {  Encapsulation Scheme = Encapsulation.PL\_CDR\_LE;  int bufferSize = 16 + 4 + 4 + PL\_CDRHeaderSize;  ClassWithLocator v1 = new ClassWithLocator() { Locator = new Locator(IPAddress.Parse("FF00:4501:0:0:0:0:0:32"), 2700) };  SerializedPayload payload = new SerializedPayload();  payload.DataEncapsulation = EncapsulationManager.Serialize<ClassWithLocator>(v1, Scheme);  IoBuffer buffer = IoBuffer.Wrap(payload.DataEncapsulation.SerializedPayload);  Assert.AreEqual(bufferSize, buffer.Remaining);  Assert.AreEqual("00 03 00 00 48 00 18 00 02 00 00 00 8C 0A 00 00 FF 00 45 01 00 00 00 00 00 00 00 00 00 00 00 32 01 00 00 00", buffer.GetHexDump());  ClassWithLocator v2 = EncapsulationManager.Deserialize<ClassWithLocator>(buffer);  Assert.AreEqual(v1.Locator, v2.Locator);  } |
| ***Salida*** | Nombre de la prueba: *TestLocatorIpV6PL\_CDR\_LE2*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00.1546355 |

Tabla ‑. TestLocatorFromSample1PL\_CDR\_LE

|  |  |
| --- | --- |
| **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 el *ClassWithLocator* con el *Serializador* y el *Deserealizador.* |
| ***Entrada*** | Inicialmente no existe ninguna instancia del *DataEncapsulation.* |
| ***Referencia*** | 9.4.3.1 |
| ***Código*** | [TestMethod]  public void TestLocatorFromSample1PL\_CDR\_LE()  {  Encapsulation Scheme = Encapsulation.PL\_CDR\_LE;  int bufferSize = 16 + 4 + 4 + PL\_CDRHeaderSize;  ClassWithLocator v1 = new ClassWithLocator() { Locator = new Locator(IPAddress.Parse("172.16.0.128"), 36945) };  SerializedPayload payload = new SerializedPayload();  payload.DataEncapsulation = EncapsulationManager.Serialize<ClassWithLocator>(v1, Scheme);  IoBuffer buffer = IoBuffer.Wrap(payload.DataEncapsulation.SerializedPayload);  Assert.AreEqual(bufferSize, buffer.Remaining);  Assert.AreEqual("00 03 00 00 48 00 18 00 01 00 00 00 51 90 00 00 00 00 00 00 00 00 00 00 00 00 00 00 AC 10 00 80 01 00 00 00", buffer.GetHexDump());  ClassWithLocator v2 = EncapsulationManager.Deserialize<ClassWithLocator>(buffer);  Assert.AreEqual(v1.Locator, v2.Locator);  } |
| ***Salida*** | Nombre de la prueba: *TestLocatorFromSample1PL\_CDR\_LE*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00.1686305 |

Tabla ‑. TestLocatorFromSample2PL\_CDR\_LE

|  |  |
| --- | --- |
| **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 el *ClassWithLocator* con el *Serializador* y el *Deserealizador.* |
| ***Entrada*** | Inicialmente no existe ninguna instancia del *DataEncapsulation.* |
| ***Referencia*** | 9.4.3.1 |
| ***Código*** | [TestMethod]  public void TestLocatorFromSample2PL\_CDR\_LE()  {  Encapsulation Scheme = Encapsulation.PL\_CDR\_LE;  int bufferSize = 16 + 4 + 4 + PL\_CDRHeaderSize;  ClassWithLocator v1 = new ClassWithLocator() { Locator = new Locator(IPAddress.Parse("239.255.0.1"), 9652) };  SerializedPayload payload = new SerializedPayload();  payload.DataEncapsulation = EncapsulationManager.Serialize<ClassWithLocator>(v1, Scheme);  IoBuffer buffer = IoBuffer.Wrap(payload.DataEncapsulation.SerializedPayload);  Assert.AreEqual(bufferSize, buffer.Remaining);  Assert.AreEqual("00 03 00 00 48 00 18 00 01 00 00 00 B4 25 00 00 00 00 00 00 00 00 00 00 00 00 00 00 EF FF 00 01 01 00 00 00", buffer.GetHexDump());  ClassWithLocator v2 = EncapsulationManager.Deserialize<ClassWithLocator>(buffer);  Assert.AreEqual(v1.Locator, v2.Locator);  } |
| ***Salida*** | Nombre de la prueba: *TestLocatorFromSample2PL\_CDR\_LE*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00.1738197 |

Tabla ‑. TestLocatorFromSample3PL\_CDR\_LE

|  |  |
| --- | --- |
| **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 el *ClassWithLocator* con el *Serializador* y el *Deserealizador.* |
| ***Entrada*** | Inicialmente no existe ninguna instancia del *DataEncapsulation.* |
| ***Referencia*** | 9.4.3.1 |
| ***Código*** | [TestMethod]  public void TestLocatorFromSample3PL\_CDR\_LE()  {  Encapsulation Scheme = Encapsulation.PL\_CDR\_LE;  int bufferSize = 16 + 4 + 4 + PL\_CDRHeaderSize;  ClassWithLocator v1 = new ClassWithLocator() { Locator = new Locator(IPAddress.Parse("127.0.0.1"), 12345) };  SerializedPayload payload = new SerializedPayload();  payload.DataEncapsulation = EncapsulationManager.Serialize<ClassWithLocator>(v1, Scheme);  IoBuffer buffer = IoBuffer.Wrap(payload.DataEncapsulation.SerializedPayload);  Assert.AreEqual(bufferSize, buffer.Remaining);  Assert.AreEqual("00 03 00 00 48 00 18 00 01 00 00 00 39 30 00 00 00 00 00 00 00 00 00 00 00 00 00 00 7F 00 00 01 01 00 00 00", buffer.GetHexDump());  ClassWithLocator v2 = EncapsulationManager.Deserialize<ClassWithLocator>(buffer);  Assert.AreEqual(v1.Locator, v2.Locator);  } |
| ***Salida*** | Nombre de la prueba: *TestLocatorFromSample3PL\_CDR\_LE*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00.1984551 |

Tabla ‑. TestGUIDCDR\_BE

|  |  |
| --- | --- |
| **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 el *EntityId* con el *Serializador* y el *Deserealizador.* |
| ***Entrada*** | Inicialmente no existe ninguna instancia del *DataEncapsulation.* |
| ***Referencia*** | 10.2.1 |
| ***Código*** | [public void TestGUIDCDR\_BE()  {  Encapsulation Scheme = Encapsulation.CDR\_BE;  int bufferSize = 16 + CDRHeaderSize;  GUID v1 = new GUID(new GuidPrefix(new byte[] { 0x0, 0x1, 0x2, 0x3, 0x4, 0x5, 0x6, 0x7, 0x8, 0x9, 0xA, 0xB }),  new EntityId(new byte[] { 0x0, 0x1, 0x2 }, EntityKinds.BUILT\_IN\_PARTICIPANT));  SerializedPayload payload = new SerializedPayload();  payload.DataEncapsulation = EncapsulationManager.Serialize<GUID>(v1, Scheme);  IoBuffer buffer = IoBuffer.Wrap(payload.DataEncapsulation.SerializedPayload);  Assert.AreEqual(bufferSize, buffer.Remaining);  Assert.AreEqual("00 00 00 00 00 01 02 03 04 05 06 07 08 09 0A 0B 00 01 02 C1", buffer.GetHexDump());  GUID v2 = EncapsulationManager.Deserialize<GUID>(buffer);  Assert.AreEqual(v1, v2);  } |
| ***Salida*** | Nombre de la prueba: *TestGUIDCDR\_BE*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 00:00:00.2030336 |

Tabla ‑. TestGUICDR\_LE

|  |  |
| --- | --- |
| **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 el *EntityId* con el *Serializador* y el *Deserealizador.* |
| ***Entrada*** | Inicialmente no existe ninguna instancia del *DataEncapsulation.* |
| ***Referencia*** | 10.2.1 |
| ***Código*** | [TestMethod]  public void TestGUIDCDR\_LE()  {  Encapsulation Scheme = Encapsulation.CDR\_LE;  int bufferSize = 16 + CDRHeaderSize;  GUID v1 = new GUID(new GuidPrefix(new byte[] { 0x0, 0x1, 0x2, 0x3, 0x4, 0x5, 0x6, 0x7, 0x8, 0x9, 0xA, 0xB }),  new EntityId(new byte[] { 0x0, 0x1, 0x2 }, EntityKinds.BUILT\_IN\_PARTICIPANT));  SerializedPayload payload = new SerializedPayload();  payload.DataEncapsulation = EncapsulationManager.Serialize<GUID>(v1, Scheme);  IoBuffer buffer = IoBuffer.Wrap(payload.DataEncapsulation.SerializedPayload);  Assert.AreEqual(bufferSize, buffer.Remaining);  Assert.AreEqual("00 01 00 00 00 01 02 03 04 05 06 07 08 09 0A 0B 00 01 02 C1", buffer.GetHexDump());  GUID v2 = EncapsulationManager.Deserialize<GUID>(buffer);  Assert.AreEqual(v1, v2);  } |
| ***Salida*** | Nombre de la prueba: *TestGUICDR\_LE*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00.2267873 |

Tabla ‑. TestGUIDPL\_CDR\_BE

|  |  |
| --- | --- |
| **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 el *EntityId* con el *Serializador* y el *Deserealizador.* |
| ***Entrada*** | Inicialmente no existe ninguna instancia del *DataEncapsulation.* |
| ***Referencia*** | 10.2.1 |
| ***Código*** | [TestMethod]  public void TestGUIDPL\_CDR\_BE()  {  Encapsulation Scheme = Encapsulation.PL\_CDR\_BE;  int bufferSize = 24 + CDRHeaderSize;  ClassWithGUID v1 = new ClassWithGUID()  {  Key = new GUID(new GuidPrefix(new byte[] { 0x0, 0x1, 0x2, 0x3, 0x4, 0x5, 0x6, 0x7, 0x8, 0x9, 0xA, 0xB }),  new EntityId(new byte[] { 0x0, 0x1, 0x2 }, EntityKinds.BUILT\_IN\_PARTICIPANT))  };  SerializedPayload payload = new SerializedPayload();  payload.DataEncapsulation = EncapsulationManager.Serialize<ClassWithGUID>(v1, Scheme);  IoBuffer buffer = IoBuffer.Wrap(payload.DataEncapsulation.SerializedPayload);  Assert.AreEqual(bufferSize, buffer.Remaining);  Assert.AreEqual("00 02 00 00 00 50 00 10 00 01 02 03 04 05 06 07 08 09 0A 0B 00 01 02 C1 00 01 00 00", buffer.GetHexDump());  ClassWithGUID v2 = EncapsulationManager.Deserialize<ClassWithGUID>(buffer);  Assert.AreEqual(v1.Key, v2.Key);  } |
| ***Salida*** | Nombre de la prueba: *TestGUIDPL\_CDR\_BE*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00.2057796 |

Tabla ‑. TestGUIDPL\_CDR\_LE

|  |  |
| --- | --- |
| **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 el *EntityId* con el *Serializador* y el *Deserealizador.* |
| ***Entrada*** | Inicialmente no existe ninguna instancia del *DataEncapsulation.* |
| ***Referencia*** | 10.2.1 |
| ***Código*** | [TestMethod]  public void TestGUIDPL\_CDR\_LE()  {  Encapsulation Scheme = Encapsulation.PL\_CDR\_LE;  int bufferSize = 24 + CDRHeaderSize;  ClassWithGUID v1 = new ClassWithGUID()  {  Key = new GUID(new GuidPrefix(new byte[] { 0x0, 0x1, 0x2, 0x3, 0x4, 0x5, 0x6, 0x7, 0x8, 0x9, 0xA, 0xB }),  new EntityId(new byte[] { 0x0, 0x1, 0x2 }, EntityKinds.BUILT\_IN\_PARTICIPANT))  };  SerializedPayload payload = new SerializedPayload();  payload.DataEncapsulation = EncapsulationManager.Serialize<ClassWithGUID>(v1, Scheme);  IoBuffer buffer = IoBuffer.Wrap(payload.DataEncapsulation.SerializedPayload);  Assert.AreEqual(bufferSize, buffer.Remaining);  Assert.AreEqual("00 03 00 00 50 00 10 00 00 01 02 03 04 05 06 07 08 09 0A 0B 00 01 02 C1 01 00 00 00", buffer.GetHexDump());  ClassWithGUID v2 = EncapsulationManager.Deserialize<ClassWithGUID>(buffer);  Assert.AreEqual(v1.Key, v2.Key);  } |
| ***Salida*** | Nombre de la prueba: *TestGUIDPL\_CDR\_LE*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00.2031448 |

Tabla ‑. TestVendorIdCDR\_BE

|  |  |
| --- | --- |
| **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 el *EntityId* con el *Serializador* y el *Deserealizador.* |
| ***Entrada*** | Inicialmente no existe ninguna instancia del *DataEncapsulation.* |
| ***Referencia*** | 10.2.1 |
| ***Código*** | [TestMethod]  public void TestVendorIdCDR\_BE()  {  Encapsulation Scheme = Encapsulation.CDR\_BE;  int bufferSize = 2 + CDRHeaderSize;  VendorId v1 = VendorId.Doopec;  SerializedPayload payload = new SerializedPayload();  payload.DataEncapsulation = EncapsulationManager.Serialize<VendorId>(v1, Scheme);  IoBuffer buffer = IoBuffer.Wrap(payload.DataEncapsulation.SerializedPayload);  Assert.AreEqual(bufferSize, buffer.Remaining);  Assert.AreEqual("00 00 00 00 01 0F", buffer.GetHexDump());  VendorId v2 = EncapsulationManager.Deserialize<VendorId>(buffer);  Assert.AreEqual(v1, v2);  } |
| ***Salida*** | Nombre de la prueba: *TestVendorIdCDR\_BE*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00.1845436 |

Tabla ‑. TestVendorIdCDR\_LE

|  |  |
| --- | --- |
| **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 el *EntityId* con el *Serializador* y el *Deserealizador.* |
| ***Entrada*** | Inicialmente no existe ninguna instancia del *DataEncapsulation.* |
| ***Referencia*** | 10.2.1 |
| ***Código*** | [TestMethod]  public void TestVendorIDCDR\_LE()  {  Encapsulation Scheme = Encapsulation.CDR\_LE;  int bufferSize = 2 + CDRHeaderSize;  VendorId v1 = VendorId.Doopec; ;  SerializedPayload payload = new SerializedPayload();  payload.DataEncapsulation = EncapsulationManager.Serialize<VendorId>(v1, Scheme);  IoBuffer buffer = IoBuffer.Wrap(payload.DataEncapsulation.SerializedPayload);  Assert.AreEqual(bufferSize, buffer.Remaining);  Assert.AreEqual("00 01 00 00 01 0F", buffer.GetHexDump());  VendorId v2 = EncapsulationManager.Deserialize<VendorId>(buffer);  Assert.AreEqual(v1, v2);  } |
| ***Salida*** | Nombre de la prueba: *TestVendorIdCDR\_LE*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00.1874632 |

Tabla ‑. TestProcotolVersionCDR\_BE

|  |  |
| --- | --- |
| **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 el *ProtocolVersion* con el *Serializador* y el *Deserealizador.* |
| ***Entrada*** | Inicialmente no existe ninguna instancia del *DataEncapsulation.* |
| ***Referencia*** | 10.2.1 |
| ***Código*** | [TestMethod]  public void TestProtocolVersionCDR\_BE()  {  Encapsulation Scheme = Encapsulation.CDR\_BE;  int bufferSize = 2 + CDRHeaderSize;  ProtocolVersion v1 = ProtocolVersion.PROTOCOLVERSION\_2\_1;  SerializedPayload payload = new SerializedPayload();  payload.DataEncapsulation = EncapsulationManager.Serialize<ProtocolVersion>(v1, Scheme);  IoBuffer buffer = IoBuffer.Wrap(payload.DataEncapsulation.SerializedPayload);  Assert.AreEqual(bufferSize, buffer.Remaining);  Assert.AreEqual("00 00 00 00 02 01", buffer.GetHexDump());  ProtocolVersion v2 = EncapsulationManager.Deserialize<ProtocolVersion>(buffer);  Assert.AreEqual(v1, v2);  } |
| ***Salida*** | Nombre de la prueba: *TestProcotolVersionCDR\_BE*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00.2274154 |

Tabla ‑. TestProcotolVersionCDR\_LE

|  |  |
| --- | --- |
| **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 el *ProtocolVersion* con el *Serializador* y el *Deserealizador.* |
| ***Entrada*** | Inicialmente no existe ninguna instancia del *DataEncapsulation.* |
| ***Referencia*** | 10.2.1 |
| ***Código*** | [TestMethod]  public void TestProtocolVersionCDR\_LE()  {  Encapsulation Scheme = Encapsulation.CDR\_LE;  int bufferSize = 2 + CDRHeaderSize;  ProtocolVersion v1 = ProtocolVersion.PROTOCOLVERSION\_2\_1;  SerializedPayload payload = new SerializedPayload();  payload.DataEncapsulation = EncapsulationManager.Serialize<ProtocolVersion>(v1, Scheme);  IoBuffer buffer = IoBuffer.Wrap(payload.DataEncapsulation.SerializedPayload);  Assert.AreEqual(bufferSize, buffer.Remaining);  Assert.AreEqual("00 01 00 00 02 01", buffer.GetHexDump());  ProtocolVersion v2 = EncapsulationManager.Deserialize<ProtocolVersion>(buffer);  Assert.AreEqual(v1, v2);  } |
| ***Salida*** | Nombre de la prueba: *TestProcotolVersionCDR\_LE*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00.1825883 |

Tabla ‑. TestProcotolVersionPL\_CDR\_BE

|  |  |
| --- | --- |
| **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 el *ClassWithProtocolVersion* con el *Serializador* y el *Deserealizador.* |
| ***Entrada*** | Inicialmente no existe ninguna instancia del *DataEncapsulation.* |
| ***Referencia*** | 10.2.1 |
| ***Código*** | [TestMethod]  public void TestProtocolVersionPL\_CDR\_BE()  {  Encapsulation Scheme = Encapsulation.PL\_CDR\_BE;  int bufferSize = 12 + CDRHeaderSize;  ClassWithProtocolVersion v1 = new ClassWithProtocolVersion() { ProtocolVersion = ProtocolVersion.PROTOCOLVERSION\_2\_1 };  SerializedPayload payload = new SerializedPayload();  payload.DataEncapsulation = EncapsulationManager.Serialize<ClassWithProtocolVersion>(v1, Scheme);  IoBuffer buffer = IoBuffer.Wrap(payload.DataEncapsulation.SerializedPayload);  Assert.AreEqual(bufferSize, buffer.Remaining);  Assert.AreEqual("00 02 00 00 00 15 00 04 02 01 00 00 00 01 00 00", buffer.GetHexDump());  ClassWithProtocolVersion v2 = EncapsulationManager.Deserialize<ClassWithProtocolVersion>(buffer);  Assert.AreEqual(v1.ProtocolVersion, v2.ProtocolVersion);  } |
| ***Salida*** | Nombre de la prueba: *TestProcotolVersionPL\_CDR\_BE*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00.200578 |

Tabla ‑. TestProcotolVersionPL\_CDR\_LE

|  |  |
| --- | --- |
| **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 el *ClassWithProtocolVersion* con el *Serializador* y el *Deserealizador.* |
| ***Entrada*** | Inicialmente no existe ninguna instancia del *DataEncapsulation.* |
| ***Referencia*** | 10.2.1 |
| ***Código*** | [TestMethod]  public void TestProtocolVersionPL\_CDR\_LE()  {  Encapsulation Scheme = Encapsulation.PL\_CDR\_LE;  int bufferSize = 12 + CDRHeaderSize;  ClassWithProtocolVersion v1 = new ClassWithProtocolVersion() { ProtocolVersion = ProtocolVersion.PROTOCOLVERSION\_2\_1 };  SerializedPayload payload = new SerializedPayload();  payload.DataEncapsulation = EncapsulationManager.Serialize<ClassWithProtocolVersion>(v1, Scheme);  IoBuffer buffer = IoBuffer.Wrap(payload.DataEncapsulation.SerializedPayload);  Assert.AreEqual(bufferSize, buffer.Remaining);  Assert.AreEqual("00 03 00 00 15 00 04 00 02 01 00 00 01 00 00 00", buffer.GetHexDump());  ClassWithProtocolVersion v2 = EncapsulationManager.Deserialize<ClassWithProtocolVersion>(buffer);  Assert.AreEqual(v1.ProtocolVersion, v2.ProtocolVersion);  } |
| ***Salida*** | Nombre de la prueba: *TestProcotolVersionPL\_CDR\_LE*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00.200578 |

Tabla ‑. TestListLocatorCDR\_BE

|  |  |
| --- | --- |
| **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 el *Locator* con el *Serializador* y el *Deserealizador.* |
| ***Entrada*** | Inicialmente no existe ninguna instancia del *DataEncapsulation.* |
| ***Referencia*** | 10.2.1 |
| ***Código*** | [TestMethod]  public void TestListLocatorCDR\_BE()  {  Encapsulation Scheme = Encapsulation.CDR\_BE;  int bufferSize = 16 + 4 + 4 + 4 + CDRHeaderSize;  List<Locator> v1 = new List<Locator>() { new Locator(IPAddress.Parse("10.20.30.40"), 2700) };  SerializedPayload payload = new SerializedPayload();  payload.DataEncapsulation = EncapsulationManager.Serialize<List<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 00 01 00 00 0A 8C 00 00 00 00 00 00 00 00 00 00 00 00 0A 14 1E 28", buffer.GetHexDump());  List<Locator> v2 = EncapsulationManager.Deserialize<List<Locator>>(buffer);  Assert.AreEqual(v1.Count, v2.Count);  Assert.AreEqual(v1[0], v2[0]);  } |
| ***Salida*** | Nombre de la prueba: *TestListLocatorCDR\_BE*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00.200578 |

Tabla ‑. TestListLocatorCDR\_LE

|  |  |
| --- | --- |
| **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 el *Locator* con el *Serializador* y el *Deserealizador.* |
| ***Entrada*** | Inicialmente no existe ninguna instancia del *DataEncapsulation.* |
| ***Referencia*** | 10.2.1 |
| ***Código*** | [TestMethod]  public void TestListLocatorCDR\_LE()  {  Encapsulation Scheme = Encapsulation.CDR\_LE;  int bufferSize = 16 + 4 + 4 + 4 + CDRHeaderSize;  List<Locator> v1 = new List<Locator>() { new Locator(IPAddress.Parse("10.20.30.40"), 2700) };  SerializedPayload payload = new SerializedPayload();  payload.DataEncapsulation = EncapsulationManager.Serialize<List<Locator>>(v1, Scheme);  IoBuffer buffer = IoBuffer.Wrap(payload.DataEncapsulation.SerializedPayload);  Assert.AreEqual(bufferSize, buffer.Remaining);  Assert.AreEqual("00 01 00 00 01 00 00 00 01 00 00 00 8C 0A 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0A 14 1E 28", buffer.GetHexDump());  List<Locator> v2 = EncapsulationManager.Deserialize<List<Locator>>(buffer);  Assert.AreEqual(v1.Count, v2.Count);  Assert.AreEqual(v1[0], v2[0]);  } |
| ***Salida*** | Nombre de la prueba: *TestListLocatorCDR\_LE*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00.2074474 |

#### Prueba de Mensajes

Tabla ‑. TestInfoDestination

|  |  |
| --- | --- |
| **Llamada:**  public InfoDestination(GuidPrefix guidPrefix) | |
| ***Descripción*** | En esta prueba se verifica el submensaje *InfoDestination.* |
| ***Entrada*** | Inicialmente no existe ninguna instancia del *InfoDestination.* |
| ***Referencia*** | 8.3.7.7 |
| ***Código*** | [TestMethod]  public void TestInfoDestination()  {  // Create a Message with InfoDestination  Message m1 = new Message();  m1.SubMessages.Add(new InfoDestination(GuidPrefix.GUIDPREFIX\_UNKNOWN));  // Write Message to bytes1 array  byte[] bytes1 = Write(m1);  // Read from bytes1 array - tests reading  Message m2 = Read(bytes1);  // Write the message Read to bytes2  byte[] bytes2 = Write(m2);  // Test, that bytes1 and bytes2 are equal  AssertArrayEquals(bytes1, bytes2);  } |
| ***Salida*** | Nombre de la prueba: *TestInfoDestination*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00.1653231 |

Tabla ‑. TestInfoSource

|  |  |
| --- | --- |
| **Llamada:**  public InfoSource(GuidPrefix guidPrefix) | |
| ***Descripción*** | En esta prueba se verifica el submensaje *InfoSource.* |
| ***Entrada*** | Inicialmente no existe ninguna instancia del *InfoDestination.* |
| ***Referencia*** | 8.3.7.9 |
| ***Código*** | [TestMethod]  public void TestInfoSource()  {  // Create a Message with InfoSource  Message m1 = new Message();  m1.SubMessages.Add(new InfoSource(GuidPrefix.GUIDPREFIX\_UNKNOWN));  // Write Message to bytes1 array  byte[] bytes1 = Write(m1);  // Read from bytes1 array - tests reading  Message m2 = Read(bytes1);  // Write the message Read to bytes2  byte[] bytes2 = Write(m2);  // Test, that bytes1 and bytes2 are equal  AssertArrayEquals(bytes1, bytes2);  } |
| ***Salida*** | Nombre de la prueba: *TestInfoSource*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00.1801263 |

Tabla ‑. TestInfoReply

|  |  |
| --- | --- |
| **Llamada:**  public InfoReply(IList<Locator> unicastLocators, IList<Locator> multicastLocators) | |
| ***Descripción*** | En esta prueba se verifica el submensaje *InfoReply.* |
| ***Entrada*** | Inicialmente no existe ninguna instancia del *InfoReply.* |
| ***Referencia*** | 8.3.7.8 |
| ***Código*** | [TestMethod]  public void TestInfoReply()  {  // Create a Message with InfoReply  Message m1 = new Message();  Locator loc1 = new Locator(IPAddress.Loopback, 7111);  Locator loc2 = new Locator(IPAddress.Loopback, 7222);  IList<Locator> unicastLocators = new List<Locator>();  unicastLocators.Add(loc1);  IList<Locator> multicastLocators = new List<Locator>();  multicastLocators.Add(loc2);  m1.SubMessages.Add(new InfoReply(unicastLocators, multicastLocators));  // Write Message to bytes1 array  byte[] bytes1 = Write(m1);  // Read from bytes1 array - tests reading  Message m2 = Read(bytes1);  // Write the message Read to bytes2  byte[] bytes2 = Write(m2);  // Test, that bytes1 and bytes2 are equal  AssertArrayEquals(bytes1, bytes2);  } |
| ***Salida*** | Nombre de la prueba: *TestInfoReply*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00.2084802 |

Tabla ‑. TestInfoReplyIp4

|  |  |
| --- | --- |
| **Llamada:**  public InfoReplyIp4(LocatorUDPv4 unicastLocator, LocatorUDPv4 multicastLocator) | |
| ***Descripción*** | En esta prueba se verifica el submensaje *InfoReplyIp4.* |
| ***Entrada*** | Inicialmente no existe ninguna instancia del *InfoReplyIp4.* |
| ***Referencia*** | 8.3.7.8 |
| ***Código*** | [TestMethod]  public void TestInfoReplyIp4()  {  // Create a Message with InfoReplyIp4  Message m1 = new Message();  LocatorUDPv4 lc1 = new LocatorUDPv4(IPAddress.Loopback, 7111);  LocatorUDPv4 lc2 = LocatorUDPv4.LOCATORUDPv4\_INVALID;  m1.SubMessages.Add(new InfoReplyIp4(lc1, lc2));  // Write Message to bytes1 array  byte[] bytes1 = Write(m1);  // Read from bytes1 array - tests reading  Message m2 = Read(bytes1);  // Write the message Read to bytes2  byte[] bytes2 = Write(m2);  // Test, that bytes1 and bytes2 are equal  AssertArrayEquals(bytes1, bytes2);  } |
| ***Salida*** | Nombre de la prueba: *TestInfoReplyIp4*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00.169154 |

Tabla ‑. TestInfoTimestamp

|  |  |
| --- | --- |
| **Llamada:**  public InfoTimestamp(long systemCurrentMillis) | |
| ***Descripción*** | En esta prueba se verifica el submensaje *InfoTimestamp.* |
| ***Entrada*** | Inicialmente no existe ninguna instancia del *InfoTimestamp.* |
| ***Referencia*** | 8.3.7.10 |
| ***Código*** | [TestMethod]  public void TestInfoTimestamp()  {  // Create a Message with InfoDestination  Message m1 = new Message();  m1.SubMessages.Add(new InfoTimestamp(123));  // Write Message to bytes1 array  byte[] bytes1 = Write(m1);  // Read from bytes1 array - tests reading  Message m2 = Read(bytes1);  // Write the message Read to bytes2  byte[] bytes2 = Write(m2);  // Test, that bytes1 and bytes2 are equal  AssertArrayEquals(bytes1, bytes2);  } |
| ***Salida*** | Nombre de la prueba: *TestInfoTimestamp*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00.1588509 |

Tabla ‑. TestGap

|  |  |
| --- | --- |
| **Llamada:**  public Gap() | |
| ***Descripción*** | En esta prueba se verifica el submensaje *Gap.* |
| ***Entrada*** | Inicialmente no existe ninguna instancia del *Gap.* |
| ***Referencia*** | 8.3.7.4 |
| ***Código*** | [TestMethod]  public void TestGap()  {  // Create a Message with Gap  Message m1 = new Message();  Gap gap = new Gap();  EntityId id1 = EntityId.ENTITYID\_UNKNOWN;  EntityId id2 = EntityId.ENTITYID\_UNKNOWN;  gap.ReaderId = id1;  gap.WriterId = id2;  gap.GapStart = new SequenceNumber(10);  gap.GapList = new SequenceNumberSet(10, new int[] { 12, 15, 19 });  m1.SubMessages.Add(gap);  // Write Message to bytes1 array  byte[] bytes1 = Write(m1);  // Read from bytes1 array - tests reading  Message m2 = Read(bytes1);  // Write the message Read to bytes2  byte[] bytes2 = Write(m2);  // Test, that bytes1 and bytes2 are equal  AssertArrayEquals(bytes1, bytes2);  } |
| ***Salida*** | Nombre de la prueba: *TestGap*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00.166937 |

Tabla ‑. TestAckNack

|  |  |
| --- | --- |
| **Llamada:**  public AckNack() | |
| ***Descripción*** | En esta prueba se verifica el submensaje *AckNack.* |
| ***Entrada*** | Inicialmente no existe ninguna instancia del *AckNack.* |
| ***Referencia*** | 8.3.7.1 |
| ***Código*** | [TestMethod]  public void TestAckNack()  {  // Create a Message with AckNack  Message m1 = new Message();  AckNack ackNack = new AckNack();  EntityId id1 = EntityId.ENTITYID\_UNKNOWN;  EntityId id2 = EntityId.ENTITYID\_UNKNOWN;  ackNack.ReaderId = id1;  ackNack.WriterId = id2;  ackNack.ReaderSNState = new SequenceNumberSet(10, new int[] { 12, 15, 19 });  ackNack.Count = 10;  m1.SubMessages.Add(ackNack);  // Write Message to bytes1 array  byte[] bytes1 = Write(m1);  // Read from bytes1 array - tests reading  Message m2 = Read(bytes1);  // Write the message Read to bytes2  byte[] bytes2 = Write(m2);  // Test, that bytes1 and bytes2 are equal  AssertArrayEquals(bytes1, bytes2);  } |
| ***Salida*** | Nombre de la prueba: *TestAckNack*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00.1699033 |

Tabla ‑. TestPad

|  |  |
| --- | --- |
| **Llamada:**  public Pad() | |
| ***Descripción*** | En esta prueba se verifica el submensaje *Pad.* |
| ***Entrada*** | Inicialmente no existe ninguna instancia del *Pad.* |
| ***Referencia*** | 8.3.7.12 |
| ***Código*** | [TestMethod]  public void TestPad()  {  // Create a Message with Pad  Message m1 = new Message();  Pad pad = new Pad();  pad.Bytes = new byte[] { 12, 15, 19 };  m1.SubMessages.Add(pad);  // Write Message to bytes1 array  byte[] bytes1 = Write(m1);  // Read from bytes1 array - tests reading  Message m2 = Read(bytes1);  // Write the message Read to bytes2  byte[] bytes2 = Write(m2);  // Test, that bytes1 and bytes2 are equal  AssertArrayEquals(bytes1, bytes2);  } |
| ***Salida*** | Nombre de la prueba: *TestPad*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00.1486317 |

Tabla ‑. TestHeartbeat

|  |  |
| --- | --- |
| **Llamada:**  public Heartbeat() | |
| ***Descripción*** | En esta prueba se verifica el submensaje *Heartbeat.* |
| ***Entrada*** | Inicialmente no existe ninguna instancia del *Heartbeat.* |
| ***Referencia*** | 8.3.7.6 |
| ***Código*** | [TestMethod]  public void TestHeartbeat()  {  // Create a Message with Heartbeat  Message m1 = new Message();  Heartbeat heartbeat = new Heartbeat();  EntityId id1 = EntityId.ENTITYID\_UNKNOWN;  EntityId id2 = EntityId.ENTITYID\_UNKNOWN;  heartbeat.readerId = id1;  heartbeat.writerId = id2;  heartbeat.firstSN = new SequenceNumber(10);  heartbeat.lastSN = new SequenceNumber(20);  heartbeat.count = 5;  m1.SubMessages.Add(heartbeat);  // Write Message to bytes1 array  byte[] bytes1 = Write(m1);  // Read from bytes1 array - tests reading  Message m2 = Read(bytes1);  // Write the message Read to bytes2  byte[] bytes2 = Write(m2);  // Test, that bytes1 and bytes2 are equal  AssertArrayEquals(bytes1, bytes2);  } |
| ***Salida*** | Nombre de la prueba: *TestHeartbeat*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00.1949603 |

Tabla ‑. TestNackFrag

|  |  |
| --- | --- |
| **Llamada:**  public NackFrag() | |
| ***Descripción*** | En esta prueba se verifica el submensaje *NackFrag.* |
| ***Entrada*** | Inicialmente no existe ninguna instancia del *NackFrag.* |
| ***Referencia*** | 8.3.7.11 |
| ***Código*** | [TestMethod]  public void TestNackFrag()  {  // Create a Message with Heartbeat  Message m1 = new Message();  NackFrag nackFrag = new NackFrag();  EntityId id1 = EntityId.ENTITYID\_UNKNOWN;  EntityId id2 = EntityId.ENTITYID\_UNKNOWN;  nackFrag.ReaderId = id1;  nackFrag.WriterId = id2;  nackFrag.FragmentNumberState = new SequenceNumberSet(5, new int[] { 6, 7, 21 });  nackFrag.WriterSequenceNumber = new SequenceNumber(20);  nackFrag.Count = 2;  m1.SubMessages.Add(nackFrag);  // Write Message to bytes1 array  byte[] bytes1 = Write(m1);  // Read from bytes1 array - tests reading  Message m2 = Read(bytes1);  // Write the message Read to bytes2  byte[] bytes2 = Write(m2);  // Test, that bytes1 and bytes2 are equal  AssertArrayEquals(bytes1, bytes2);  } |
| ***Salida*** | Nombre de la prueba: *TestNackFrag*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00.2558301 |

Tabla ‑. TestHeartbeatFrag

|  |  |
| --- | --- |
| **Llamada:**  public HeartbeatFrag() | |
| ***Descripción*** | En esta prueba se verifica el submensaje *HeartbeatFrag.* |
| ***Entrada*** | Inicialmente no existe ninguna instancia del *HeartbeatFrag..* |
| ***Referencia*** | 8.3.7.6 |
| ***Código*** | /// <summary>  /// Tests, that reading and writing of HeartbeatFrag is symmetrical.  /// </summary>  [TestMethod]  public void TestHeartbeatFrag()  {  // Create a Message with Heartbeat  Message m1 = new Message();  HeartbeatFrag heartbeatFrag = new HeartbeatFrag();  EntityId id1 = EntityId.ENTITYID\_UNKNOWN;  EntityId id2 = EntityId.ENTITYID\_UNKNOWN;  heartbeatFrag.ReaderId = id1;  heartbeatFrag.WriterId = id2;  heartbeatFrag.WriterSequenceNumber = new SequenceNumber(10);  heartbeatFrag.LastFragmentNumber = 30;  heartbeatFrag.Count = 50;  m1.SubMessages.Add(heartbeatFrag);  // Write Message to bytes1 array  byte[] bytes1 = Write(m1);  // Read from bytes1 array - tests reading  Message m2 = Read(bytes1);  // Write the message Read to bytes2  byte[] bytes2 = Write(m2);  // Test, that bytes1 and bytes2 are equal  AssertArrayEquals(bytes1, bytes2);  } |
| ***Salida*** | Nombre de la prueba: *TestHeartbeatFrag*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00.1587616 |

Tabla ‑. TestDataFrag

|  |  |
| --- | --- |
| **Llamada:**  public DataFrag() | |
| ***Descripción*** | En esta prueba se verifica el submensaje *DataFrag.* |
| ***Entrada*** | Inicialmente no existe ninguna instancia del *DataFrag.* |
| ***Referencia*** | 8.3.7.3 |
| ***Código*** | /// <summary>  /// Tests, that reading and writing of DataFrag is symmetrical.  /// </summary>  [TestMethod]  public void TestDataFrag()  {  // Create a Message with DataFrag  Message m1 = new Message();  DataFrag dataFrag = new DataFrag();  EntityId id1 = EntityId.ENTITYID\_UNKNOWN;  EntityId id2 = EntityId.ENTITYID\_UNKNOWN;  dataFrag.ReaderId = id1;  dataFrag.WriterId = id2;  dataFrag.WriterSequenceNumber = new SequenceNumber(10);  dataFrag.FragmentStartingNumber = 30;  dataFrag.FragmentsInSubmessage = 1;  dataFrag.FragmentSize = 4;  dataFrag.SerializedPayload = new byte[] { 100, 10, 1, 0 };  m1.SubMessages.Add(dataFrag);  // Write Message to bytes1 array  byte[] bytes1 = Write(m1);  // Read from bytes1 array - tests reading  Message m2 = Read(bytes1);  // Write the message Read to bytes2  byte[] bytes2 = Write(m2);  // Test, that bytes1 and bytes2 are equal  AssertArrayEquals(bytes1, bytes2);  } |
| ***Salida*** | Nombre de la prueba: *TestDataFrag*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:00.1957048 |

### Transporte

#### Prueba de Detección de paquetes RTPS.

Tabla ‑. TestPublishData

|  |  |
| --- | --- |
| **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*** | No aplica |
| ***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.

Tabla ‑. TestPublishPacket2

|  |  |
| --- | --- |
| **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= |

Tabla ‑. GeneralRTPSMessageTesterMethod

|  |  |
| --- | --- |
| **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*** | No aplica |
| ***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 |

Tabla 4‑46. TestOpenDDS\_rtps\_reliability\_runtest\_localPacket01

|  |  |
| --- | --- |
| **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*** | No aplica |
| ***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 |

Tabla 4‑47. TestOpenDDS\_rtps\_reliability\_runtest\_localPacket02

|  |  |
| --- | --- |
| **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*** | No aplica |
| ***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 |

Tabla 4‑48. TestOpenDDS\_rtps\_reliability\_runtest\_localPacket03

|  |  |
| --- | --- |
| **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*** | No aplica |
| ***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*** | No aplica |
| ***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 |

Tabla ‑. TestGeneration2

|  |  |
| --- | --- |
| **Llamada:**  static GuidGenerator() | |
| ***Descripción*** | En esta prueba se verifica el correcto funcionamiento del Guid Generator. |
| ***Entrada*** | Inicialmente no se tiene inicializado al *GuidGenerator* |
| ***Referencia*** | No aplica |
| ***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.

Tabla ‑. TestWorkerVerySlow

|  |  |
| --- | --- |
| **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*** | No aplica |
| ***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 |

Tabla ‑. TestWorkerSlow

|  |  |
| --- | --- |
| **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*** | No aplica |
| ***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 |

Tabla ‑. TestWorkerQuick

|  |  |
| --- | --- |
| **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*** | No aplica |
| ***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

Tabla ‑. TestTimeSeconds

|  |  |
| --- | --- |
| **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*** | No aplica |
| ***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

Tabla ‑. TestParticipantBuiltinTopicData

|  |  |
| --- | --- |
| **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*** | 9.3.2 |
| ***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 |

Tabla ‑. TestPublicationBuiltinTopicData

|  |  |
| --- | --- |
| **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*** | 9.3.2 |
| ***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 |

Tabla ‑. TestSubscriptionBuiltinTopicData

|  |  |
| --- | --- |
| **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*** | 9.3.2 |
| ***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 |

Tabla ‑. TestSubscriptionBuiltinTopicData

|  |  |
| --- | --- |
| **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*** | 9.3.2 |
| ***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*** | 9.3.2 |
| ***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.

Tabla ‑. TestBoolPacketLE

|  |  |
| --- | --- |
| **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*** | 9.2.2 |
| ***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 |

Tabla ‑. TestCharPacketLE

|  |  |
| --- | --- |
| **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*** | 9.2.2 |
| ***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 |

Tabla ‑. TestU8PacketLE

|  |  |
| --- | --- |
| **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*** | 9.2.2 |
| ***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 |

Tabla ‑. TestU16PacketLE

|  |  |
| --- | --- |
| **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*** | 9.2.2 |
| ***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 |

Tabla ‑. TestU32PacketLE

|  |  |
| --- | --- |
| **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*** | 10.2.1 |
| ***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 |

Tabla ‑. TestU64PacketLE

|  |  |
| --- | --- |
| **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*** | 10.2.1 |
| ***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 |

Tabla ‑. TestS8PacketLE1

|  |  |
| --- | --- |
| **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*** | 10.2.1 |
| ***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 |

Tabla ‑. TestS8PacketLE2

|  |  |
| --- | --- |
| **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*** | 10.2.1 |
| ***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 |

Tabla ‑. TestS16PacketLE1

|  |  |
| --- | --- |
| **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*** | 10.2.1 |
| ***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 |

Tabla ‑. TestS16PacketLE2

|  |  |
| --- | --- |
| **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*** | 10.2.1 |
| ***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 |

Tabla ‑. TestS32PacketLE1

|  |  |
| --- | --- |
| **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*** | 10.2.1 |
| ***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 |

Tabla ‑. TestS32PacketLE2

|  |  |
| --- | --- |
| **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*** | 10.2.1 |
| ***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 |

Tabla ‑. TestS64PacketLE1

|  |  |
| --- | --- |
| **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*** | 10.2.1 |
| ***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 |

Tabla ‑. TestS64PacketLE2

|  |  |
| --- | --- |
| **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*** | 10.2.1 |
| ***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 |

Tabla ‑. TestSinglePacketLE

|  |  |
| --- | --- |
| **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*** | 10.2.1 |
| ***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 |

Tabla ‑. TestDoublePacketLE

|  |  |
| --- | --- |
| **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*** | 10.2.1 |
| ***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 |

Tabla ‑. TestBoolSequencePacketLE

|  |  |
| --- | --- |
| **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*** | 10.2.1 |
| ***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 |

Tabla ‑. TestShortSequencePacketLE

|  |  |
| --- | --- |
| **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*** | 10.2.1 |
| ***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 |

Tabla ‑. TestEnumSequencePacketLE

|  |  |
| --- | --- |
| **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*** | 10.2.1 |
| ***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 |

Tabla ‑. TestIntSequencePacketLE

|  |  |
| --- | --- |
| **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*** | 10.2.1 |
| ***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 |

Tabla ‑. TestEmptyBoolSequencePacketLE

|  |  |
| --- | --- |
| **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*** | 10.2.1 |
| ***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 |

Tabla ‑. TestEmptyShortSequencePacketLE

|  |  |
| --- | --- |
| **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*** | 10.2.1 |
| ***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 |

Tabla ‑. TestEmptyEnumSequencePacketLE

|  |  |
| --- | --- |
| **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*** | 10.2.1 |
| ***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 |

Tabla ‑. TestEmptyIntSequencePacketLE

|  |  |
| --- | --- |
| **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*** | 10.2.1 |
| ***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 |

Tabla ‑. TestBoolPacketBE

|  |  |
| --- | --- |
| **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*** | 10.2.1 |
| ***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 |

Tabla ‑. TestCharPacketBE

|  |  |
| --- | --- |
| **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*** | 10.2.1 |
| ***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 |

Tabla ‑. TestU8PackeBE

|  |  |
| --- | --- |
| **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*** | 10.2.1 |
| ***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 |

Tabla ‑. TestU16PacketBE

|  |  |
| --- | --- |
| **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*** | 10.2.1 |
| ***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 |

Tabla ‑. TestU32PacketBE

|  |  |
| --- | --- |
| **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*** | 10.2.1 |
| ***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 |

Tabla ‑. TestU64PacketBE

|  |  |
| --- | --- |
| **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*** | 10.2.1 |
| ***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 |

Tabla ‑. TestS8PacketBE1

|  |  |
| --- | --- |
| **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*** | 10.2.1 |
| ***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 |

Tabla ‑. TestS8PacketBE2

|  |  |
| --- | --- |
| **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*** | 10.2.1 |
| ***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 |

Tabla ‑. TestS16PacketBE1

|  |  |
| --- | --- |
| **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*** | 10.2.1 |
| ***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 |

Tabla ‑. TestS16PacketBE2

|  |  |
| --- | --- |
| **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*** | 10.2.1 |
| ***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 |

Tabla ‑. TestS32PacketBE1

|  |  |
| --- | --- |
| **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*** | 10.2.1 |
| ***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 |

Tabla ‑. TestS32PacketBE2

|  |  |
| --- | --- |
| **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*** | 10.2.1 |
| ***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 |

Tabla ‑. TestS64PacketBE1

|  |  |
| --- | --- |
| **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*** | 10.2.1 |
| ***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 |

Tabla ‑. TestS64PacketLE2

|  |  |
| --- | --- |
| **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*** | 10.2.1 |
| ***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 |

Tabla ‑. TestSinglePacketBE

|  |  |
| --- | --- |
| **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*** | 10.2.1 |
| ***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 |

Tabla ‑. TestDoublePacketLE

|  |  |
| --- | --- |
| **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*** | 10.2.1 |
| ***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 |

Tabla ‑. TestBoolSequencePacketBE

|  |  |
| --- | --- |
| **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*** | 10.2.1 |
| ***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 |

Tabla ‑. TestShortSequencePacketBE

|  |  |
| --- | --- |
| **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*** | 10.2.1 |
| ***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 |

Tabla ‑. TestEnumSequencePacketBE

|  |  |
| --- | --- |
| **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*** | 10.2.1 |
| ***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 |

Tabla ‑. TestIntSequencePacketBE

|  |  |
| --- | --- |
| **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*** | 10.2.1 |
| ***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 |

Tabla ‑. TestEmptyBoolSequencePacketBE

|  |  |
| --- | --- |
| **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*** | 10.2.1 |
| ***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 |

Tabla ‑. TestEmptyShortSequencePacketBE

|  |  |
| --- | --- |
| **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*** | 10.2.1 |
| ***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 |

Tabla ‑. TestEmptyEnumSequencePacketBE

|  |  |
| --- | --- |
| **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*** | 10.2.1 |
| ***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 |

Tabla ‑. TestEmptyIntSequencePacketLE

|  |  |
| --- | --- |
| **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*** | 10.2.1 |
| ***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.

Tabla ‑. TestExploreMyClass1

|  |  |
| --- | --- |
| **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*** | No aplica |
| ***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.

Tabla ‑. TestBoolPacket

|  |  |
| --- | --- |
| **Llamada:**  public BoolPacket(bool v) | |
| ***Descripción*** | En esta prueba se verifica que se está serializando y deserializando correctamente el *BoolPacket.* |
| ***Entrada*** | En la prueba no se tiene inicializado el fichero. |
| ***Referencia*** | No aplica |
| ***Código*** | [TestMethod]  public void TestBoolPacket()  {  BoolPacket v1 = new BoolPacket(true);  var buffer = ByteBufferAllocator.Instance.Allocate(sizeof(bool));  Serializer.Serialize(buffer, v1);  Assert.AreEqual(sizeof(bool), buffer.Position);  buffer.Rewind();  BoolPacket v2 = Serializer.Deserialize<BoolPacket>(buffer);  Assert.AreEqual(v1, v2);  Assert.AreEqual(sizeof(bool), buffer.Position);  } |
| ***Salida*** | Nombre de la prueba: *TestBoolPacket*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 0:00:39.9332615 |

Tabla ‑. TestCharPacket

|  |  |
| --- | --- |
| **Llamada:**  public CharPacket(char v) | |
| ***Descripción*** | En esta prueba se verifica que se está serializando y deserializando correctamente el *CharPacket.* |
| ***Entrada*** | En la prueba no se tiene inicializado el fichero. |
| ***Referencia*** | No aplica |
| ***Código*** | [TestMethod]  public void TestCharPacket()  {  CharPacket v1 = new CharPacket('A');  var buffer = ByteBufferAllocator.Instance.Allocate(sizeof(char));  Serializer.Serialize(buffer, v1);  Assert.AreEqual(sizeof(char), buffer.Position);  buffer.Rewind();  CharPacket v2 = Serializer.Deserialize<CharPacket>(buffer);  Assert.AreEqual(v1, v2);  Assert.AreEqual(sizeof(char), buffer.Position);  } |
| ***Salida*** | Nombre de la prueba: *TestCharPacket*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: < 1 ms |

Tabla ‑. TestU8Packet

|  |  |
| --- | --- |
| **Llamada:**  public U8Packet(byte v) | |
| ***Descripción*** | En esta prueba se verifica que se está serializando y deserializando correctamente el *U8Packet.* |
| ***Entrada*** | En la prueba no se tiene inicializado el fichero. |
| ***Referencia*** | No aplica |
| ***Código*** | [TestMethod]  public void TestU8Packet()  {  U8Packet v1 = new U8Packet(0xA);  var buffer = ByteBufferAllocator.Instance.Allocate(sizeof(byte));  Serializer.Serialize(buffer, v1);  Assert.AreEqual(sizeof(byte), buffer.Position);  buffer.Rewind();  U8Packet v2 = Serializer.Deserialize<U8Packet>(buffer);  Assert.AreEqual(v1, v2);  Assert.AreEqual(sizeof(byte), buffer.Position);  } |
| ***Salida*** | Nombre de la prueba: *TestU8Packet*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: < 1 ms |

Tabla ‑. TestU16Packet

|  |  |
| --- | --- |
| **Llamada:**  public U16Packet(ushort v) | |
| ***Descripción*** | En esta prueba se verifica que se está serializando y deserializando correctamente el *U16Packet.* |
| ***Entrada*** | En la prueba no se tiene inicializado el fichero. |
| ***Referencia*** | No aplica |
| ***Código*** | [TestMethod]  public void TestU16Packet()  {  U16Packet v1 = new U16Packet(0xAB);  var buffer = ByteBufferAllocator.Instance.Allocate(sizeof(ushort));  Serializer.Serialize(buffer, v1);  Assert.AreEqual(sizeof(ushort), buffer.Position);  buffer.Rewind();  U16Packet v2 = Serializer.Deserialize<U16Packet>(buffer);  Assert.AreEqual(v1, v2);  Assert.AreEqual(sizeof(ushort), buffer.Position);  } |
| ***Salida*** | Nombre de la prueba: *TestU16Packet*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: < 1 ms |

Tabla ‑. TestU32Packet

|  |  |
| --- | --- |
| **Llamada:**  public U32Packet(uint v) | |
| ***Descripción*** | En esta prueba se verifica que se está serializando y deserializando correctamente el *U32Packet.* |
| ***Entrada*** | En la prueba no se tiene inicializado el fichero. |
| ***Referencia*** | No aplica |
| ***Código*** | [TestMethod]  public void TestU32Packet()  {  U32Packet v1 = new U32Packet(0xABA);  var buffer = ByteBufferAllocator.Instance.Allocate(sizeof(uint));  Serializer.Serialize(buffer, v1);  Assert.AreEqual(sizeof(uint), buffer.Position);  buffer.Rewind();  U32Packet v2 = Serializer.Deserialize<U32Packet>(buffer);  Assert.AreEqual(v1, v2);  Assert.AreEqual(sizeof(uint), buffer.Position);  } |
| ***Salida*** | Nombre de la prueba: *TestU32Packet*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: < 1 ms |

Tabla ‑. TestU64Packet

|  |  |
| --- | --- |
| **Llamada:**  public U64Packet(ulong v) | |
| ***Descripción*** | En esta prueba se verifica que se está serializando y deserializando correctamente el *U64Packet.* |
| ***Entrada*** | En la prueba no se tiene inicializado el fichero. |
| ***Referencia*** | No aplica |
| ***Código*** | [TestMethod]  public void TestU64Packet()  {  U64Packet v1 = new U64Packet(0xABCD);  var buffer = ByteBufferAllocator.Instance.Allocate(sizeof(ulong));  Serializer.Serialize(buffer, v1);  Assert.AreEqual(sizeof(ulong), buffer.Position);  buffer.Rewind();  U64Packet v2 = Serializer.Deserialize<U64Packet>(buffer);  Assert.AreEqual(v1, v2);  Assert.AreEqual(sizeof(ulong), buffer.Position);  } |
| ***Salida*** | Nombre de la prueba: *TestU64Packet*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: < 1 ms |

Tabla ‑. TestS8Packet

|  |  |
| --- | --- |
| **Llamada:**  public S8Packet(sbyte v) | |
| ***Descripción*** | En esta prueba se verifica que se está serializando y deserializando correctamente el *S8Packet.* |
| ***Entrada*** | En la prueba no se tiene inicializado el fichero. |
| ***Referencia*** | No aplica |
| ***Código*** | [TestMethod]  public void TestS8Packet()  {  S8Packet v1 = new S8Packet(0xA);  var buffer = ByteBufferAllocator.Instance.Allocate(sizeof(sbyte));  Serializer.Serialize(buffer, v1);  Assert.AreEqual(sizeof(sbyte), buffer.Position);  buffer.Rewind();  S8Packet v2 = Serializer.Deserialize<S8Packet>(buffer);  Assert.AreEqual(v1, v2);  Assert.AreEqual(sizeof(sbyte), buffer.Position);  } |
| ***Salida*** | Nombre de la prueba: *TestS8Packet*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: < 1 ms |

Tabla ‑. TestS16Packet

|  |  |
| --- | --- |
| **Llamada:**  public S16Packet(short v) | |
| ***Descripción*** | En esta prueba se verifica que se está serializando y deserializando correctamente el *S16Packet.* |
| ***Entrada*** | En la prueba no se tiene inicializado el fichero. |
| ***Referencia*** | No aplica |
| ***Código*** | [TestMethod]  public void TestS16Packet()  {  S16Packet v1 = new S16Packet(0xAB);  var buffer = ByteBufferAllocator.Instance.Allocate(sizeof(short));  Serializer.Serialize(buffer, v1);  Assert.AreEqual(sizeof(short), buffer.Position);  buffer.Rewind();  S16Packet v2 = Serializer.Deserialize<S16Packet>(buffer);  Assert.AreEqual(v1, v2);  Assert.AreEqual(sizeof(short), buffer.Position);  } |
| ***Salida*** | Nombre de la prueba: *TestS16Packet*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: < 1 ms |

Tabla ‑. TestS32Packet

|  |  |
| --- | --- |
| **Llamada:**  public S32Packet(int v) | |
| ***Descripción*** | En esta prueba se verifica que se está serializando y deserializando correctamente el *S32Packet.* |
| ***Entrada*** | En la prueba no se tiene inicializado el fichero. |
| ***Referencia*** | No aplica |
| ***Código*** | [TestMethod]  public void TestS32Packet()  {  S32Packet v1 = new S32Packet(0xABA);  var buffer = ByteBufferAllocator.Instance.Allocate(sizeof(int));  Serializer.Serialize(buffer, v1);  Assert.AreEqual(sizeof(int), buffer.Position);  buffer.Rewind();  S32Packet v2 = Serializer.Deserialize<S32Packet>(buffer);  Assert.AreEqual(v1, v2);  Assert.AreEqual(sizeof(int), buffer.Position);  } |
| ***Salida*** | Nombre de la prueba: *TestS32Packet*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: < 1 ms |

Tabla ‑. TestS64Packet

|  |  |
| --- | --- |
| **Llamada:**  public S64Packet(long v) | |
| ***Descripción*** | En esta prueba se verifica que se está serializando y deserializando correctamente el *S64Packet.* |
| ***Entrada*** | En la prueba no se tiene inicializado el fichero. |
| ***Referencia*** | No aplica |
| ***Código*** | [TestMethod]  public void TestS64Packet()  {  S64Packet v1 = new S64Packet(0xABCD);  var buffer = ByteBufferAllocator.Instance.Allocate(sizeof(long));  Serializer.Serialize(buffer, v1);  Assert.AreEqual(sizeof(long), buffer.Position);  buffer.Rewind();  S64Packet v2 = Serializer.Deserialize<S64Packet>(buffer);  Assert.AreEqual(v1, v2);  Assert.AreEqual(sizeof(long), buffer.Position);  } |
| ***Salida*** | Nombre de la prueba: *TestS64Packet*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: < 1 ms |

Tabla ‑. TestSinglePacket

|  |  |
| --- | --- |
| **Llamada:**  public SinglePacket(float v) | |
| ***Descripción*** | En esta prueba se verifica que se está serializando y deserializando correctamente el *SinglePacket.* |
| ***Entrada*** | En la prueba no se tiene inicializado el fichero. |
| ***Referencia*** | No aplica |
| ***Código*** | [TestMethod]  public void TestSinglePacket()  {  SinglePacket v1 = new SinglePacket(0.1f);  var buffer = ByteBufferAllocator.Instance.Allocate(sizeof(float));  Serializer.Serialize(buffer, v1);  Assert.AreEqual(sizeof(float), buffer.Position);  buffer.Rewind();  SinglePacket v2 = Serializer.Deserialize<SinglePacket>(buffer);  Assert.AreEqual(v1, v2);  Assert.AreEqual(sizeof(float), buffer.Position);  } |
| ***Salida*** | Nombre de la prueba: *TestSinglePacket*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: < 1 ms |

Tabla ‑. TestDoublePacket

|  |  |
| --- | --- |
| **Llamada:**  public DoublePacket(double v) | |
| ***Descripción*** | En esta prueba se verifica que se está serializando y deserializando correctamente el *DoublePacket.* |
| ***Entrada*** | En la prueba no se tiene inicializado el fichero. |
| ***Referencia*** | No aplica |
| ***Código*** | [TestMethod]  public void TestDoublePacket()  {  DoublePacket v1 = new DoublePacket(0.1);  var buffer = ByteBufferAllocator.Instance.Allocate(sizeof(double));  Serializer.Serialize(buffer, v1);  Assert.AreEqual(sizeof(double), buffer.Position);  buffer.Rewind();  DoublePacket v2 = Serializer.Deserialize<DoublePacket>(buffer);  Assert.AreEqual(v1, v2);  Assert.AreEqual(sizeof(double), buffer.Position);  } |
| ***Salida*** | Nombre de la prueba: *TestDoublePacket*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: < 1 ms |

Tabla ‑. TestPrimitivesPacket

|  |  |
| --- | --- |
| **Llamada:**  public PrimitivesPacket(int seed) | |
| ***Descripción*** | En esta prueba se verifica que se está serializando y deserializando correctamente el *PrimitivesPacket.* |
| ***Entrada*** | En la prueba no se tiene inicializado el fichero. |
| ***Referencia*** | No aplica |
| ***Código*** | [TestMethod]  public void TestPrimitivesPacket()  {  int size = PrimitivesPacket.Size();  PrimitivesPacket v1 = new PrimitivesPacket(15);  var buffer = ByteBufferAllocator.Instance.Allocate(size);  Serializer.Serialize(buffer, v1);  Assert.AreEqual(size, buffer.Position);  buffer.Rewind();  PrimitivesPacket v2 = Serializer.Deserialize<PrimitivesPacket>(buffer);  Assert.AreEqual(v1, v2);  Assert.AreEqual(size, buffer.Position);  } |
| ***Salida*** | Nombre de la prueba: *TestPrimitivesPacket*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 1 ms |

Tabla ‑. TestEnumPacket

|  |  |
| --- | --- |
| **Llamada:**  public EnumPacket(MyEnum v) | |
| ***Descripción*** | En esta prueba se verifica que se está serializando y deserializando correctamente el *EnumPacket.* |
| ***Entrada*** | En la prueba no se tiene inicializado el fichero. |
| ***Referencia*** | No aplica |
| ***Código*** | [TestMethod]  public void TestEnumPacket()  {  int size = EnumPacket.Size();  EnumPacket v1 = new EnumPacket(MyEnum.Three);  var buffer = ByteBufferAllocator.Instance.Allocate(size);  Serializer.Serialize(buffer, v1);  Assert.AreEqual(size, buffer.Position);  buffer.Rewind();  EnumPacket v2 = Serializer.Deserialize<EnumPacket>(buffer);  Assert.AreEqual(v1, v2);  Assert.AreEqual(size, buffer.Position);  } |
| ***Salida*** | Nombre de la prueba: *TestEnumPacket*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: < 1 ms |

Tabla ‑. TestStruct1Packet

|  |  |
| --- | --- |
| **Llamada:**  public struct MyStruct1 | |
| ***Descripción*** | En esta prueba se verifica que se está serializando y deserializando correctamente el *MyStruct1.* |
| ***Entrada*** | En la prueba no se tiene inicializado el fichero. |
| ***Referencia*** | No aplica |
| ***Código*** | [TestMethod]  public void TestStruct1Packet()  {  int size = MyStruct1.Size();  MyStruct1 v1 = new MyStruct1();  v1.m\_byte = 1;  v1.m\_int = 2;  v1.m\_long = 3;  var buffer = ByteBufferAllocator.Instance.Allocate(size);  Serializer.Serialize(buffer, v1);  Assert.AreEqual(size, buffer.Position);  buffer.Rewind();  MyStruct1 v2 = Serializer.Deserialize<MyStruct1>(buffer);  Assert.AreEqual(v1, v2);  Assert.AreEqual(size, buffer.Position);  } |
| ***Salida*** | Nombre de la prueba: *TestStruct1Packet*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 1 ms |

Tabla ‑. TestStructMessagePacket

|  |  |
| --- | --- |
| **Llamada:**  public StructMessage() | |
| ***Descripción*** | En esta prueba se verifica que se está serializando y deserializando correctamente el *StructMessage.* |
| ***Entrada*** | En la prueba no se tiene inicializado el fichero. |
| ***Referencia*** | No aplica |
| ***Código*** | [TestMethod]  public void TestStructMessagePacket()  {  int size = StructMessage.Size();  StructMessage v1 = new StructMessage();  var buffer = ByteBufferAllocator.Instance.Allocate(size);  Serializer.Serialize(buffer, v1);  Assert.AreEqual(size, buffer.Position);  buffer.Rewind();  StructMessage v2 = Serializer.Deserialize<StructMessage>(buffer);  Assert.AreEqual(v1, v2);  Assert.AreEqual(size, buffer.Position);  } |
| ***Salida*** | Nombre de la prueba: *TestStructMessagePacket*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 1 ms |

Tabla ‑. TestSequenceMessagePacket

|  |  |
| --- | --- |
| **Llamada:**  public IntSequencePacket(int[] v) | |
| ***Descripción*** | En esta prueba se verifica que se está serializando y deserializando correctamente el *IntSequencePacket.* |
| ***Entrada*** | En la prueba no se tiene inicializado el fichero. |
| ***Referencia*** | No aplica |
| ***Código*** | [TestMethod]  public void TestSequenceMessagePacket()  {  IntSequencePacket v1 = new IntSequencePacket(new int[] { 1, 2, 3 });  int size = (1 + v1.m\_val.Length) \* sizeof(int);  var buffer = ByteBufferAllocator.Instance.Allocate(size);  Serializer.Serialize(buffer, v1);  Assert.AreEqual(size, buffer.Position);  buffer.Rewind();  IntSequencePacket v2 = Serializer.Deserialize<IntSequencePacket>(buffer);  Assert.AreEqual(v1, v2);  Assert.AreEqual(size, buffer.Position);  } |
| ***Salida*** | Nombre de la prueba: *TestSequenceMessagePacket*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: < 1 ms |

Tabla ‑. TestSequenceMessagePacket2

|  |  |
| --- | --- |
| **Llamada:**  public IntSequencePacket(int[] v) | |
| ***Descripción*** | En esta prueba se verifica que se está serializando y deserializando correctamente el *IntSequencePacket.* |
| ***Entrada*** | En la prueba no se tiene inicializado el fichero. |
| ***Referencia*** | No aplica |
| ***Código*** | [TestMethod]  public void TestSequenceMessagePacket2()  {  IntSequencePacket v1 = new IntSequencePacket(new int[] { });  int size = (1 + v1.m\_val.Length) \* sizeof(int);  var buffer = ByteBufferAllocator.Instance.Allocate(size);  Serializer.Serialize(buffer, v1);  Assert.AreEqual(size, buffer.Position);  buffer.Rewind();  IntSequencePacket v2 = Serializer.Deserialize<IntSequencePacket>(buffer);  Assert.AreEqual(v1, v2);  Assert.AreEqual(size, buffer.Position);  } |
| ***Salida*** | Nombre de la prueba: *TestSequenceMessagePacket2*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: < 1 ms |

Tabla ‑. TestSequenceMessagePacket3

|  |  |
| --- | --- |
| **Llamada:**  public IntSequencePacket(int[] v) | |
| ***Descripción*** | En esta prueba se verifica que se está serializando y deserializando correctamente el *IntSequencePacket.* |
| ***Entrada*** | En la prueba no se tiene inicializado el fichero. |
| ***Referencia*** | No aplica |
| ***Código*** | [TestMethod]  public void TestSequenceMessagePacket3()  {  IntSequencePacket v1 = new IntSequencePacket(null);  int size = (1) \* sizeof(int);  var buffer = ByteBufferAllocator.Instance.Allocate(size);  Serializer.Serialize(buffer, v1);  Assert.AreEqual(size, buffer.Position);  buffer.Rewind();  IntSequencePacket v2 = Serializer.Deserialize<IntSequencePacket>(buffer);  Assert.AreEqual(v1, v2);  Assert.AreEqual(size, buffer.Position);  } |
| ***Salida*** | Nombre de la prueba: *TestSequenceMessagePacket3*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: < 1 ms |

Tabla ‑. TestSequenceMessagePacket3

|  |  |
| --- | --- |
| **Llamada:**  public IntSequencePacket(int[] v) | |
| ***Descripción*** | En esta prueba se verifica que se está serializando y deserializando correctamente el *IntSequencePacket.* |
| ***Entrada*** | En la prueba no se tiene inicializado el fichero. |
| ***Referencia*** | No aplica |
| ***Código*** | [TestMethod]  public void TestSequenceMessagePacket3()  {  IntSequencePacket v1 = new IntSequencePacket(null);  int size = (1) \* sizeof(int);  var buffer = ByteBufferAllocator.Instance.Allocate(size);  Serializer.Serialize(buffer, v1);  Assert.AreEqual(size, buffer.Position);  buffer.Rewind();  IntSequencePacket v2 = Serializer.Deserialize<IntSequencePacket>(buffer);  Assert.AreEqual(v1, v2);  Assert.AreEqual(size, buffer.Position);  } |
| ***Salida*** | Nombre de la prueba: *TestSequenceMessagePacket3*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: < 1 ms |

Tabla ‑. TestMyClassListMessagePacket1

|  |  |
| --- | --- |
| **Llamada:**  public class MyClassList | |
| ***Descripción*** | En esta prueba se verifica que está serializando y deserializando correctamente la clase *MyClassList.* |
| ***Entrada*** | En la prueba no se tiene inicializado el fichero. |
| ***Referencia*** | No aplica |
| ***Código*** | [TestMethod]  public void TestMyClassListMessagePacket1()  {  MyClassList v1 = new MyClassList();  v1.m\_intlist = new List<int>() { 5, 6, 7 };  int size = (v1.m\_intlist.Count + 1) \* 4 + 1 \* 4;  var buffer = ByteBufferAllocator.Instance.Allocate(size);  Serializer.Serialize(buffer, v1);  Assert.AreEqual(size, buffer.Position);  buffer.Rewind();  MyClassList v2 = Serializer.Deserialize<MyClassList>(buffer);  Assert.AreEqual(v1, v2);  Assert.AreEqual(size, buffer.Position);  } |
| ***Salida*** | Nombre de la prueba: *TestMyClassListMessagePacket1*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 2 ms |

Tabla ‑. TestMyClassListMessagePacket2

|  |  |
| --- | --- |
| **Llamada:**  public class MyClassList | |
| ***Descripción*** | En esta prueba se verifica que está serializando y deserializando correctamente la clase *MyClassList.* |
| ***Entrada*** | En la prueba no se tiene inicializado el fichero. |
| ***Referencia*** | No aplica |
| ***Código*** | [TestMethod]  public void TestMyClassListMessagePacket2()  {  MyClassList v1 = new MyClassList();  v1.m\_intlist = new List<int>();  int size = (v1.m\_intlist.Count + 1) \* 4 + 1 \* 4;  var buffer = ByteBufferAllocator.Instance.Allocate(size);  Serializer.Serialize(buffer, v1);  Assert.AreEqual(size, buffer.Position);  buffer.Rewind();  MyClassList v2 = Serializer.Deserialize<MyClassList>(buffer);  Assert.AreEqual(v1, v2);  Assert.AreEqual(size, buffer.Position);  } |
| ***Salida*** | Nombre de la prueba: *TestMyClassListMessagePacket2*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: < 1 ms |

Tabla ‑. TestMyClassListMessagePacket3

|  |  |
| --- | --- |
| **Llamada:**  public class MyClassList | |
| ***Descripción*** | En esta prueba se verifica que está serializando y deserializando correctamente la clase *MyClassList.* |
| ***Entrada*** | En la prueba no se tiene inicializado el fichero. |
| ***Referencia*** | No aplica |
| ***Código*** | [TestMethod]  public void TestMyClassListMessagePacket3()  {  MyClassList v1 = new MyClassList();  v1.m\_intlist = null;  int size = 1 \* 4 + 1 \* 4;  var buffer = ByteBufferAllocator.Instance.Allocate(size);  Serializer.Serialize(buffer, v1);  Assert.AreEqual(size, buffer.Position);  buffer.Rewind();  MyClassList v2 = Serializer.Deserialize<MyClassList>(buffer);  Assert.IsNull(v1.m\_intlist);  Assert.IsNull(v2.m\_intlist);  Assert.AreEqual(v1.m\_int, v2.m\_int);  Assert.AreEqual(size, buffer.Position);  } |
| ***Salida*** | Nombre de la prueba: *TestMyClassListMessagePacket3*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: < 1 ms |

#### Pruebas de Primitivas.

Tabla ‑. TestDouble

|  |  |
| --- | --- |
| **Llamada:**  public struct Double : IComparable, IFormattable, IConvertible, IComparable<double>, IEquatable<double> | |
| ***Descripción*** | En esta prueba se verifica el estado de la primitiva *doublé.* |
| ***Entrada*** | En la prueba no se tiene inicializado el fichero. |
| ***Referencia*** | No aplica |
| ***Código*** | [TestMethod]  public void TestDouble()  {  double v1 = 1.0;  var buffer = ByteBufferAllocator.Instance.Allocate(sizeof(double));  Doopec.Serializer.Primitives.WritePrimitive(buffer, v1);  buffer.Rewind();  double v2;  Doopec.Serializer.Primitives.ReadPrimitive(buffer, out v2);  Assert.AreEqual(v1, v2);  } |
| ***Salida*** | Nombre de la prueba: *TestDouble*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: < 1 ms |

Tabla ‑. TestSingle

|  |  |
| --- | --- |
| **Llamada:**  public struct Single : IComparable, IFormattable, IConvertible, IComparable<float>, IEquatable<float> | |
| ***Descripción*** | En esta prueba se verifica el estado de la primitiva *float.* |
| ***Entrada*** | En la prueba no se tiene inicializado el fichero. |
| ***Referencia*** | No aplica |
| ***Código*** | [TestMethod]  public void TestSingle()  {  float v1 = 1.0f;  var buffer = ByteBufferAllocator.Instance.Allocate(sizeof(float));  Doopec.Serializer.Primitives.WritePrimitive(buffer, v1);  buffer.Rewind();  float v2;  Doopec.Serializer.Primitives.ReadPrimitive(buffer, out v2);  Assert.AreEqual(v1, v2);  } |
| ***Salida*** | Nombre de la prueba: *TestSingle*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: < 1 ms |

Tabla ‑. TestShort

|  |  |
| --- | --- |
| **Llamada:**  public struct Int16 : IComparable, IFormattable, IConvertible, IComparable<short>, IEquatable<short> | |
| ***Descripción*** | En esta prueba se verifica el estado de la primitiva *short.* |
| ***Entrada*** | En la prueba no se tiene inicializado el fichero. |
| ***Referencia*** | No aplica |
| ***Código*** | [TestMethod]  public void TestShort()  {  short v1 = 1;  var buffer = ByteBufferAllocator.Instance.Allocate(sizeof(short));  Doopec.Serializer.Primitives.WritePrimitive(buffer, v1);  buffer.Rewind();  short v2;  Doopec.Serializer.Primitives.ReadPrimitive(buffer, out v2);  Assert.AreEqual(v1, v2);  } |
| ***Salida*** | Nombre de la prueba: *TestShort*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: < 1 ms |

Tabla ‑. TestUShort

|  |  |
| --- | --- |
| **Llamada:**  public struct UInt16 : IComparable, IFormattable, IConvertible, IComparable<ushort>, IEquatable<ushort> | |
| ***Descripción*** | En esta prueba se verifica el estado de la primitiva *UShort.* |
| ***Entrada*** | En la prueba no se tiene inicializado el fichero. |
| ***Referencia*** | No aplica |
| ***Código*** | [TestMethod]  public void TestUShort()  {  ushort v1 = 1;  var buffer = ByteBufferAllocator.Instance.Allocate(sizeof(ushort));  Doopec.Serializer.Primitives.WritePrimitive(buffer, v1);  buffer.Rewind();  ushort v2;  Doopec.Serializer.Primitives.ReadPrimitive(buffer, out v2);  Assert.AreEqual(v1, v2);  } |
| ***Salida*** | Nombre de la prueba: *TestUShort*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: < 1 ms |

Tabla ‑. TestInt

|  |  |
| --- | --- |
| **Llamada:**  public struct Int32 : IComparable, IFormattable, IConvertible, IComparable<int>, IEquatable<int> | |
| ***Descripción*** | En esta prueba se verifica el estado de la primitiva *int.* |
| ***Entrada*** | En la prueba no se tiene inicializado el fichero. |
| ***Referencia*** | No aplica |
| ***Código*** | [TestMethod]  public void TestInt()  {  int v1 = 1;  var buffer = ByteBufferAllocator.Instance.Allocate(sizeof(int));  Doopec.Serializer.Primitives.WritePrimitive(buffer, v1);  buffer.Rewind();  int v2;  Doopec.Serializer.Primitives.ReadPrimitive(buffer, out v2);  Assert.AreEqual(v1, v2);  } |
| ***Salida*** | Nombre de la prueba: *TestInt*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: < 1 ms |

Tabla ‑. TestUInt

|  |  |
| --- | --- |
| **Llamada:**  public struct UInt32 : IComparable, IFormattable, IConvertible, IComparable<uint>, IEquatable<uint> | |
| ***Descripción*** | En esta prueba se verifica el estado de la primitiva *uint.* |
| ***Entrada*** | En la prueba no se tiene inicializado el fichero. |
| ***Referencia*** | No aplica |
| ***Código*** | [TestMethod]  public void TestUInt()  {  uint v1 = 1;  var buffer = ByteBufferAllocator.Instance.Allocate(sizeof(uint));  Doopec.Serializer.Primitives.WritePrimitive(buffer, v1);  buffer.Rewind();  uint v2;  Doopec.Serializer.Primitives.ReadPrimitive(buffer, out v2);  Assert.AreEqual(v1, v2);  } |
| ***Salida*** | Nombre de la prueba: *TestUInt*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: < 1 ms |

Tabla ‑. TestLong

|  |  |
| --- | --- |
| **Llamada:**  public struct Int64 : IComparable, IFormattable, IConvertible, IComparable<long>, IEquatable<long> | |
| ***Descripción*** | En esta prueba se verifica el estado de la primitiva *long.* |
| ***Entrada*** | En la prueba no se tiene inicializado el fichero. |
| ***Referencia*** | No aplica |
| ***Código*** | [TestMethod]  public void TestLong()  {  long v1 = 1;  var buffer = ByteBufferAllocator.Instance.Allocate(sizeof(long));  Doopec.Serializer.Primitives.WritePrimitive(buffer, v1);  buffer.Rewind();  long v2;  Doopec.Serializer.Primitives.ReadPrimitive(buffer, out v2);  Assert.AreEqual(v1, v2);  } |
| ***Salida*** | Nombre de la prueba: *TestLong*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: < 1 ms |

Tabla ‑. TestULong

|  |  |
| --- | --- |
| **Llamada:**  public struct UInt64 : IComparable, IFormattable, IConvertible, IComparable<ulong>, IEquatable<ulong> | |
| ***Descripción*** | En esta prueba se verifica el estado de la primitiva *ulong.* |
| ***Entrada*** | En la prueba no se tiene inicializado el fichero. |
| ***Referencia*** | No aplica |
| ***Código*** | [TestMethod]  public void TestULong()  {  ulong v1 = 1;  var buffer = ByteBufferAllocator.Instance.Allocate(sizeof(ulong));  Doopec.Serializer.Primitives.WritePrimitive(buffer, v1);  buffer.Rewind();  ulong v2;  Doopec.Serializer.Primitives.ReadPrimitive(buffer, out v2);  Assert.AreEqual(v1, v2);  } |
| ***Salida*** | Nombre de la prueba: *TestULong*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: < 1 ms |

Tabla ‑. TestChar

|  |  |
| --- | --- |
| **Llamada:**  public struct Char : IComparable, IConvertible, IComparable<char>, IEquatable<char> | |
| ***Descripción*** | En esta prueba se verifica el estado de la primitiva *char.* |
| ***Entrada*** | En la prueba no se tiene inicializado el fichero. |
| ***Referencia*** | No aplica |
| ***Código*** | [TestMethod]  public void TestChar()  {  char v1 = 'a';  var buffer = ByteBufferAllocator.Instance.Allocate(sizeof(char));  Doopec.Serializer.Primitives.WritePrimitive(buffer, v1);  buffer.Rewind();  char v2;  Doopec.Serializer.Primitives.ReadPrimitive(buffer, out v2);  Assert.AreEqual(v1, v2);  } |
| ***Salida*** | Nombre de la prueba: *TestChar*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: < 1 ms |

Tabla ‑. TestByte

|  |  |
| --- | --- |
| **Llamada:**  public struct Byte : IComparable, IFormattable, IConvertible, IComparable<byte>, IEquatable<byte> | |
| ***Descripción*** | En esta prueba se verifica el estado de la primitiva *byte.* |
| ***Entrada*** | En la prueba no se tiene inicializado el fichero. |
| ***Referencia*** | No aplica |
| ***Código*** | [TestMethod]  public void TestByte()  {  byte v1 = 1;  var buffer = ByteBufferAllocator.Instance.Allocate(sizeof(byte));  Doopec.Serializer.Primitives.WritePrimitive(buffer, v1);  buffer.Rewind();  byte v2;  Doopec.Serializer.Primitives.ReadPrimitive(buffer, out v2);  Assert.AreEqual(v1, v2);  } |
| ***Salida*** | Nombre de la prueba: *TestByte*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: < 1 ms |

Tabla ‑. TestSByte

|  |  |
| --- | --- |
| **Llamada:**  public struct SByte : IComparable, IFormattable, IConvertible, IComparable<sbyte>, IEquatable<sbyte> | |
| ***Descripción*** | En esta prueba se verifica el estado de la primitiva *sbyte.* |
| ***Entrada*** | En la prueba no se tiene inicializado el fichero. |
| ***Referencia*** | No aplica |
| ***Código*** | [TestMethod]  public void TestSByte()  {  sbyte v1 = -1;  var buffer = ByteBufferAllocator.Instance.Allocate(sizeof(sbyte));  Doopec.Serializer.Primitives.WritePrimitive(buffer, v1);  buffer.Rewind();  sbyte v2;  Doopec.Serializer.Primitives.ReadPrimitive(buffer, out v2);  Assert.AreEqual(v1, v2);  } |
| ***Salida*** | Nombre de la prueba: *TestSByte*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: < 1 ms |

Tabla ‑. TestBool

|  |  |
| --- | --- |
| **Llamada:**  public struct Boolean : IComparable, IConvertible, IComparable<bool>, IEquatable<bool> | |
| ***Descripción*** | En esta prueba se verifica el estado de la primitiva *bool.* |
| ***Entrada*** | En la prueba no se tiene inicializado el fichero. |
| ***Referencia*** | No aplica |
| ***Código*** | [TestMethod]  public void TestBool()  {  bool v1 = true;  var buffer = ByteBufferAllocator.Instance.Allocate(sizeof(bool));  Doopec.Serializer.Primitives.WritePrimitive(buffer, v1);  buffer.Rewind();  bool v2;  Doopec.Serializer.Primitives.ReadPrimitive(buffer, out v2);  Assert.AreEqual(v1, v2);  } |
| ***Salida*** | Nombre de la prueba: *TestBool*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: < 1 ms |

Tabla ‑. TestDateTime

|  |  |
| --- | --- |
| **Llamada:**  public struct DateTime : IComparable, IFormattable, IConvertible, ISerializable, IComparable<DateTime>, IEquatable<DateTime> | |
| ***Descripción*** | En esta prueba se verifica el estado de la primitiva *DateTime.* |
| ***Entrada*** | En la prueba no se tiene inicializado el fichero. |
| ***Referencia*** | No aplica |
| ***Código*** | [TestMethod]  public void TestDateTime()  {  DateTime v1 = DateTime.Now;  var buffer = ByteBufferAllocator.Instance.Allocate(sizeof(long));  Doopec.Serializer.Primitives.WritePrimitive(buffer, v1);  buffer.Rewind();  DateTime v2;  Doopec.Serializer.Primitives.ReadPrimitive(buffer, out v2);  Assert.AreEqual(v1, v2);  } |
| ***Salida*** | Nombre de la prueba: *TestDateTime*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 1 ms |

Tabla ‑. TestString

|  |  |
| --- | --- |
| **Llamada:**  public sealed class String : IComparable, ICloneable, IConvertible, IComparable<string>, IEnumerable<char>, IEnumerable, IEquatable<string> | |
| ***Descripción*** | En esta prueba se verifica el estado de la primitiva *string.* |
| ***Entrada*** | En la prueba no se tiene inicializado el fichero. |
| ***Referencia*** | No aplica |
| ***Código*** | [TestMethod]  public void TestString()  {  string v1 = "this is a string!";  var buffer = ByteBufferAllocator.Instance.Allocate(1 + v1.Length \* sizeof(char));  Doopec.Serializer.Primitives.WritePrimitive(buffer, v1);  buffer.Rewind();  string v2;  Doopec.Serializer.Primitives.ReadPrimitive(buffer, out v2);  Assert.AreEqual(v1, v2);  } |
| ***Salida*** | Nombre de la prueba: *TestString*  Resultado de la prueba: https://i-msdn.sec.s-msft.com/dynimg/IC689871.png  Duración de la prueba: 2 ms |