Halyri - Udp

1.0.1

Generated by Doxygen 1.8.7

Wed Jun 25 2014 00:38:35

Contents

1 Namespace Documentation

1.1 Package SimpleUdpMediaClient

Namespaces

- package Collections
- package Packets

Classes

- class UdpMediaClientSocket

  The class is client socket implementing a simple UDP-based media transfer protocol. Delivery of packets is not guaranteed. Sent packets are split to conform to the specified MTU. The underlaying IP checksum is relied upon for the integrity of the packets. The constant ping packets are employed to monitor the connection and to keep the UDP packets routed in both directions by the network channel.

Functions

- delegate void MediaPacketReceived (object sender, MediaPacket p)

  The function is delegate for media packet receive events.
- delegate void ControlPacketReceived (object sender, ControlPacket c)

  The function is delegate for control packet receive events.
- delegate void SuspectedConnectionProblem (object sender)

  The function is delegate for connection problem events.
- delegate void ConnectionFailed (object sender)

  The function is delegate for connection failure events.

1.1.1 Function Documentation

1.1.1.1 delegate void SimpleUdpMediaClient.ConnectionFailed (object *sender*)

The function is delegate for connection failure events.

Parameters

sender	The sending instance of SimpleUdpMediaClient.

1.1.1.2 delegate void SimpleUdpMediaClient.ControlPacketReceived (object *sender,* **ControlPacket** *c*)

The function is delegate for control packet receive events.

Parameters

sender	The sending instance of SimpleUdpMediaClient.
c	The received ControlPacket.

1.1.1.3 delegate void SimpleUdpMediaClient.MediaPacketReceived (object *sender,* **MediaPacket** *p*)

The function is delegate for media packet receive events.

Parameters

sender	The sending SimpleUdpMediaClient instance.
p	The received MediaPacket.

1.1.1.4 delegate void SimpleUdpMediaClient.SuspectedConnectionProblem (object *sender*)

The function is delegate for connection problem events.

Parameters

sender	The sending instance of SimpleUdpMediaClient.

1.2 Package SimpleUdpMediaClient.Collections

Classes

- class BlockingQueue< T >

  The class is a thread-safe blocking queue that has a defined, finite length.

1.3 Package SimpleUdpMediaClient.Packets

Classes

- class ControlPacket

  The class represents a control packet for the protocol defined and implement by the UdpMediaClientSocket. The packet consist of a packet type identifier, a sender GUID, a Int64 sequence number, a target GUID and a command.
- interface INetworkPacket

  The class interface represents a network level packet. It exposes methods for converting the packet from and to a byte array representation for network transfer.
- class MediaContinuationPacket

  The class represents a subsequent portion of MediaPacket. MediaContinuationPackets are used to transfer the data that remains after the data worth first MTU is taken from the MediaPacket and costructed into a MediaHeaderPacket. The combination of a MediaHeaderPacket and the subsequent MediaContinuationPackets is a transfer sequence.

- class MediaHeaderPacket

  The class represents media information and payload data up to one network MTU in length from a MediaPacket. MediaHeaderPackets are used to transfer a portion of payload data and the media information from a MediaPacket. The combination of a MediaHeaderPacket and the subsequent MediaContinuationPackets is a transfer sequence.
- class MediaInformation

  The class for media type information. It is used to the format and the compression of a media.
- class MediaPacket

  The class representts a user-level media packet. The packet is used when receiving and sending media using the UdpMediaClientSocket. It encapsulates the complete media data and media type information. It can provide a collection of NetworkPackets conforming to the network MTU for network transfer. It can later be reconstructed from such packets at the receiving end.
- class PingPacket

  The class represents a ping packet for the protocol defined and implemented by the UdpMediaClientSocket. The packet consist of a packet type identifier, a sender GUID and a Int64 sequence number.

Enumerations

- enum MediaFormat : short {
  H264, **Wav**, **Wma**, **Wmv**,
  Speex, **Opus**, **Jpg** }

  The Media format enumeration to specify media formats.
- enum TransferCompression : short { **None**, **GZip** }

  The media transfer compression format enumeration specify the used compression.

1.3.1 Enumeration Type Documentation

1.3.1.1 enum MediaFormat : short

The Media format enumeration to specify media formats.

1.3.1.2 enum TransferCompression : short

The media transfer compression format enumeration specify the used compression.

2 Class Documentation

2.1 BlockingQueue< T > Class Template Reference

The class is a thread-safe blocking queue that has a defined, finite length.

Public Member Functions

- BlockingQueue ()

  The function initializes a new blocking queue that has the default length of 30 elements.
- BlockingQueue (int capacity)

  The function initializes a blocking queue with the specified length.
- void Enqueue (T element)

  The function enqueues the element. If the queue is full, the operation will block until free space becomes available.
- bool Offer (T element)

  The function tries to enqueue the element. If the operation succeeds without blocking, it returns true. Otherwise it return false.
- T Dequeue ()

The function dequeues and returns the last element from the end of the queue. The opeartion will block if the queue is empty.

- bool Peek (out T element)

  The function tries to dequeue the last element from the end of the queue. If operation succeeds without blocking, it returns true. Otherwise it returns false.

- int Count ()

  The function gets the count of elements in the queue.

2.1.1 Detailed Description

The class is a thread-safe blocking queue that has a defined, finite length.

<author>Veli-Mikko Puupponen</author>

Template Parameters

T	The type of the elements in the queue.

2.1.2 Constructor & Destructor Documentation

2.1.2.1 BlockingQueue ( )

The function initializes a new blocking queue that has the default length of 30 elements.

2.1.2.2 BlockingQueue (int *capacity*)

The function initializes a blocking queue with the specified length.

Parameters

capacity	The length of the queue.

2.1.3 Member Function Documentation

2.1.3.1 int Count ( )

The function gets the count of elements in the queue.

Returns

The number of elements in this queue.

2.1.3.2 T Dequeue ( )

The function dequeues and returns the last element from the end of the queue. The opeartion will block if the queue is empty.

Returns

The last element in the queue.

2.1.3.3 void Enqueue (T *element*)

The function enqueues the element. If the queue is full, the operation will block until free space becomes available.

Parameters

element	The element to be enqueued.

2.1.3.4 bool Offer (T *element*)

The function tries to enqueue the element. If the operation succeeds without blocking, it returns true. Otherwise it return false.

Parameters

element	The element to be enqueued.

Returns

True, if the element was successfully enqueued, otherwise false.

2.1.3.5 bool Peek (out T *element*)

The function tries to dequeue the last element from the end of the queue. If operation succeeds without blocking, it returns true. Otherwise it returns false.

Parameters

element	The last element in the queue.

Returns

True, if the opeation succeeds, otherwise false.

The documentation for this class was generated from the following file:

- BlockingQueue.cs

2.2 ControlPacket Class Reference

The class represents a control packet for the protocol defined and implement by the UdpMediaClientSocket. The packet consist of a packet type identifier, a sender GUID, a Int64 sequence number, a target GUID and a command.

Inherits INetworkPacket.

Public Types

- enum CommandType : byte { **RequestEnableRouting**, **RequestDisableRouting**, **Statistics** }

  The enumeration for command type. It is used to specify the command of the control packet.

Public Member Functions

- ControlPacket ()

  The function initializes a new ControlPacket instance with no parameters.
- ControlPacket (string sourceGuid, string targetGuid, CommandType command, byte[] payload, Int64 sequenceNumber)

  The function initializes a new ControlPacket instance that has the specified source and target Guids, command, payload data and sequence number.
- ControlPacket (string sourceGuid, string targetGuid, CommandType command, Int64 sequenceNumber)

  The function initializes a new ControlPacket instance that has the specified source and target Guids, command and sequence number.

- ControlPacket (string sourceGuid, string targetGuid, CommandType command, byte[] payload)

  The function initializes a new ControlPacket instance that has the specified source and target Guids, command and payload data.
- ControlPacket (string sourceGuid, string targetGuid, CommandType command)

  The function initializes a new ControlPacket instance that has the specified source and target Guids and command.
- ControlPacket (byte[] sourceGuid, byte[] targetGuid, CommandType command, byte[] payload)

  The function initializes a new ControlPacket instance that has the specified source and target Guids, command and payload data.
- void FromBytes (byte[] packetBytes)

  The function gets the command, the target guid and the packet control data from the given data and saves them.
- byte[] GetBytes (long sequenceNumber)

  The function saves the given sequence number and returns the data header of the currently saved sequence.
- byte[] GetBytes ()

  The function returns the data header of the currently saved sequence.

Public Attributes

- const byte PacketId = 2

  The specified id of the packet type. It is used to identify a packet.
- const int HeaderLengthInOctets = 42

  The specified length of the packet header in octets.

Properties

- string SourceGuid `[get]`

  The function to get the Guid identifying the sender.
- string TargetGuid `[get]`

  The function to get the Guid identifying the target.
- Int64 Sequence `[get, set]`

  The function to set and get the packet sequence number.
- CommandType Command `[get]`

  The function to get the packet command.
- byte[] PayloadData `[get]`

  The function to get the packet payload data. Can be null.

2.2.1 Detailed Description

The class represents a control packet for the protocol defined and implement by the UdpMediaClientSocket. The packet consist of a packet type identifier, a sender GUID, a Int64 sequence number, a target GUID and a command.

<author>Veli-Mikko Puupponen</author> Control packets are primarily user to request server packet routing from one client to another.

This packet can have optional payload data.

2.2.2 Member Enumeration Documentation

2.2.2.1 enum CommandType : byte

The enumeration for command type. It is used to specify the command of the control packet.

2.2.3 Constructor & Destructor Documentation

2.2.3.1 ControlPacket ( )

The function initializes a new ControlPacket instance with no parameters.

2.2.3.2 ControlPacket (string *sourceGuid,* string *targetGuid,* CommandType *command,* byte[] *payload,* Int64 *sequenceNumber*)

The function initializes a new ControlPacket instance that has the specified source and target Guids, command, payload data and sequence number.

It throws an ArgumentException if source or target Guid is malformed.

Parameters

sourceGuid	The string representation of the sender's Guid.
sourceGuid	The string representation of the target's Guid.
command	The packet command.
payload	The optional payload data.
sequence↩ Number	Packet sequence number.

2.2.3.3 ControlPacket (string *sourceGuid,* string *targetGuid,* CommandType *command,* Int64 *sequenceNumber*)

The function initializes a new ControlPacket instance that has the specified source and target Guids, command and sequence number.

It throws an ArgumentException if source or target Guid is malformed.

Parameters

sourceGuid	The string representation of the sender's Guid.
sourceGuid	The string representation of the target's Guid.
command	The packet command.
sequence↩ Number	Packet sequence number.

2.2.3.4 ControlPacket (string *sourceGuid,* string *targetGuid,* CommandType *command,* byte[] *payload*)

The function initializes a new ControlPacket instance that has the specified source and target Guids, command and payload data.

It throws an ArgumentException if source or target Guid is malformed.

Parameters

sourceGuid	The string representation of the sender's Guid.
sourceGuid	The string representation of the target's Guid.
command	The packet command.
payload	The optional payload data.

2.2.3.5 ControlPacket (string *sourceGuid,* string *targetGuid,* CommandType *command*)

The function initializes a new ControlPacket instance that has the specified source and target Guids and command.

It throws an ArgumentException if source or target Guid is malformed.

Parameters

sourceGuid	The byte representation of the sender's Guid.
sourceGuid	The byte representation of the target's Guid.
command	The packet command.

2.2.3.6 ControlPacket (byte[] *sourceGuid,* byte[] *targetGuid,* CommandType *command,* byte[] *payload*)

The function initializes a new ControlPacket instance that has the specified source and target Guids, command and payload data.

It throws an ArgumentException if any of the arguments is null.

Parameters

sourceGuid	The byte representation of the sender's Guid.
sourceGuid	The byte representation of the target's Guid.
command	The packet command.
payload	The optional payload data.

2.2.4 Member Function Documentation

2.2.4.1 void FromBytes (byte[] *packetBytes*)

The function gets the command, the target guid and the packet control data from the given data and saves them.

Parameters

packetBytes	The given data.

Implements INetworkPacket.

2.2.4.2 byte [] GetBytes (long *sequenceNumber*)

The function saves the given sequence number and returns the data header of the currently saved sequence.

Returns

The data header of current the sequence.

2.2.4.3 byte [] GetBytes ( )

The function returns the data header of the currently saved sequence.

Returns

The data header of current the sequence.

Implements INetworkPacket.

2.2.5 Member Data Documentation

2.2.5.1 const int HeaderLengthInOctets = 42

The specified length of the packet header in octets.

2.2.5.2 const byte PacketId = 2

The specified id of the packet type. It is used to identify a packet.

2.2.6 Property Documentation

2.2.6.1 CommandType Command [get]

The function to get the packet command.

2.2.6.2 byte [] PayloadData [get]

The function to get the packet payload data. Can be null.

2.2.6.3 Int64 Sequence [get], [set]

The function to set and get the packet sequence number.

2.2.6.4 string SourceGuid [get]

The function to get the Guid identifying the sender.

2.2.6.5 string TargetGuid [get]

The function to get the Guid identifying the target.

The documentation for this class was generated from the following file:

- ControlPacket.cs

2.3 INetworkPacket Interface Reference

The class interface represents a network level packet. It exposes methods for converting the packet from and to a byte array representation for network transfer.

Inherited by ControlPacket, MediaContinuationPacket, MediaHeaderPacket, and PingPacket.

Public Member Functions

- byte[] GetBytes (Int64 sequence)

  The function converts this packet to a byte array for network transfer. The specified seqeuence number will be used in the packet header.
- byte[] GetBytes ()

  The function converts this packet to a byte array for network transfer. The sequence number specified in the packet will be used.
- void FromBytes (byte[] packetBytes)

  The function sets the state of this packet from the provided array of bytes. If the array does not containt a valid byte representation of the type of packet, It throws an ArgumentException.

2.3.1 Detailed Description

The class interface represents a network level packet. It exposes methods for converting the packet from and to a byte array representation for network transfer.

<author>Veli-Mikko Puupponen</author>

2.3.2 Member Function Documentation

2.3.2.1 void FromBytes (byte[] *packetBytes*)

The function sets the state of this packet from the provided array of bytes. If the array does not containt a valid byte representation of the type of packet, It throws an ArgumentException.

It throws an ArgumentException if the provided array does not containt a valid byte repsentation of the packet type.

Parameters

packetBytes	The byte representation of a packet with the same type as this packet.

Implemented in MediaHeaderPacket, ControlPacket, MediaContinuationPacket, and PingPacket.

2.3.2.2 byte [] GetBytes (Int64 *sequence*)

The function converts this packet to a byte array for network transfer. The specified seqeuence number will be used in the packet header.

It throws an ArgumentException if the packet is not in a properly initializes state.

Parameters

sequence	The packet sequence number for the transfer header.

Returns

A byte array containing data and transfer header for the packet.

2.3.2.3 byte [] GetBytes ( )

The function converts this packet to a byte array for network transfer. The sequence number specified in the packet will be used.

It throws an ArgumentException if the packet is not in a properly initializes state.

Returns

A byte array containing data and transfer header for this packet.

Implemented in ControlPacket, MediaHeaderPacket, MediaContinuationPacket, and PingPacket.

The documentation for this interface was generated from the following file:

- INetworkPacket.cs

2.4 MediaContinuationPacket Class Reference

The class represents a subsequent portion of MediaPacket. MediaContinuationPackets are used to transfer the data that remains after the data worth first MTU is taken from the MediaPacket and costructed into a MediaHeaderPacket. The combination of a MediaHeaderPacket and the subsequent MediaContinuationPackets is a transfer sequence.

Inherits INetworkPacket.

Public Member Functions

- MediaContinuationPacket ()

  The function initializes a new MediaContinuationPacket with no parameters.
- MediaContinuationPacket (byte[] guid, Int32 packetOrderNumber, Int32 packetCount, byte[] payload, Int64 sequenceNumber)

  The function initializes a new MediaContinuationPacket instance that has the specified source Guid, packet order number, packet count, payload data and sequence number.
- MediaContinuationPacket (byte[] guid, Int32 packetOrderNumber, Int32 packetCount, byte[] payload)

  The function initializes a new MediaContinuationPacket instance that has the specified source Guid, packet order number, packet count and payload data. The sequence number must be provided prior any calls to the parameterless GetBytes.
- byte[] GetBytes (long sequenceNumber)

  The function saves the given sequence number and returns the data header of the currently saved sequence.

- void FromBytes (byte[] packetBytes)

  The function gets the sequence, the total amount of packets, the packet number and the target guid from the header of the given data and saves them.
- byte[] GetBytes ()

  The function returns the data header of the currently saved sequence.

Public Attributes

- const int HeaderLengthInOctets = 33

  The specified length of the packet header in octets.
- const byte PacketId = 1

  The specified id of the packet type. It is used to identify a packet.

Properties

- string SourceGuid `[get]`

  The function to get the Guid identifying the sender.
- Int64 Sequence `[get, set]`

  The function to get and set the packet sequence number.
- Int32 TotalPacketCount `[get]`

  The function to get the total number of packets in the related transfer sequence.
- Int32 PacketNumber `[get]`

  The function to get the number of this packet in the related transfer sequence.
- byte[] PayloadData `[get]`

  The function to get the payload data.

2.4.1 Detailed Description

The class represents a subsequent portion of MediaPacket. MediaContinuationPackets are used to transfer the data that remains after the data worth first MTU is taken from the MediaPacket and costructed into a MediaHeaderPacket. The combination of a MediaHeaderPacket and the subsequent MediaContinuationPackets is a transfer sequence.

<author>Veli-Mikko Puupponen</author> This packet consist of a packet type identifier, sender GUID, a Int64 sequence number, packet count and packet number in the transfer sequence.

This packet must always have payload data.

2.4.2 Constructor & Destructor Documentation

2.4.2.1 MediaContinuationPacket ( )

The function initializes a new MediaContinuationPacket with no parameters.

2.4.2.2 MediaContinuationPacket (byte[] *guid,* Int32 *packetOrderNumber,* Int32 *packetCount,* byte[] *payload,* Int64 *sequenceNumber*)

The function initializes a new MediaContinuationPacket instance that has the specified source Guid, packet order number, packet count, payload data and sequence number.

The first continuation packet in a network level transfer sequence always has packetOrderNumber = 1.

Parameters

guid	The string representation of the sender's Guid.
packetOrderNumber	The number of the this packet in the transfer sequence.
packetCount	The total count of packets in the transfer sequence.
payload	The payload data for this packet.
sequenceNumber	The sequence number for this packet. Must not vary within a transfer sequence.

2.4.2.3 MediaContinuationPacket (byte[] *guid,* Int32 *packetOrderNumber,* Int32 *packetCount,* byte[] *payload*)

The function initializes a new MediaContinuationPacket instance that has the specified source Guid, packet order number, packet count and payload data. The sequence number must be provided prior any calls to the parameterless GetBytes.

The first continuation packet in a network level transfer sequence always has packetOrderNumber = 1.

Parameters

guid	The string representation of the sender's Guid.
packetOrderNumber	The number of the this packet in the transfer sequence.
packetCount	The total count of packets in the transfer sequence.
payload	The payload data for this packet.

2.4.3 Member Function Documentation

2.4.3.1 void FromBytes (byte[] *packetBytes*)

The function gets the sequence, the total amount of packets, the packet number and the target guid from the header of the given data and saves them.

Parameters

packetBytes	The given data.

Implements INetworkPacket.

2.4.3.2 byte [] GetBytes (long *sequenceNumber*)

The function saves the given sequence number and returns the data header of the currently saved sequence.

Returns

The data header of current the sequence.

2.4.3.3 byte [] GetBytes ( )

The function returns the data header of the currently saved sequence.

Returns

The data header of current the sequence.

Implements INetworkPacket.

2.4.4 Member Data Documentation

2.4.4.1 const int HeaderLengthInOctets = 33

The specified length of the packet header in octets.

2.4.4.2 const byte PacketId = 1

The specified id of the packet type. It is used to identify a packet.

2.4.5 Property Documentation

2.4.5.1 Int32 PacketNumber [get]

The function to get the number of this packet in the related transfer sequence.

2.4.5.2 byte [] PayloadData [get]

The function to get the payload data.

2.4.5.3 Int64 Sequence [get], [set]

The function to get and set the packet sequence number.

2.4.5.4 string SourceGuid [get]

The function to get the Guid identifying the sender.

2.4.5.5 Int32 TotalPacketCount [get]

The function to get the total number of packets in the related transfer sequence.

The documentation for this class was generated from the following file:

- MediaContinuationPacket.cs

2.5 MediaHeaderPacket Class Reference

The class represents media information and payload data up to one network MTU in length from a MediaPacket. MediaHeaderPackets are used to transfer a portion of payload data and the media information from a MediaPacket. The combination of a MediaHeaderPacket and the subsequent MediaContinuationPackets is a transfer sequence.

Inherits INetworkPacket.

Public Member Functions

- MediaHeaderPacket ()

  The function initializes a new MediaHeaderPacket instance with no parameters.
- MediaHeaderPacket (byte[] guid, Int16 mediaTypeIdentifier, Int16 mediaTransferCompressioIdentifier, Int32 packetCount, Int32 totalPayloadDataLength, Int32 originatingDataLength, byte[] payload, Int64 sequence↩ Number)

  The function initializes a new MediaHeaderPacket instance that has the specified source Guid, media type identified, media transfer compression identified, packet count, total payload data length, originating data length, sequence number and payload data
- MediaHeaderPacket (byte[] guid, Int16 mediaTypeIdentifier, Int16 mediaTransferCompressioIdentifier, Int32 packetCount, Int32 totalPayloadDataLength, Int32 originatingDataLength, byte[] payload)

  The function initializes a new MediaHeaderPacket instance that has the specified source Guid, media type identified, media transfer compression identified, packet count, total payload data length, originating data length and payload data
- byte[] GetBytes (long sequenceNumber)

The function saves the given sequence number and returns the data header of the given sequence.

- void FromBytes (byte[] packetBytes)

  The function gets the media data, the packet control data and the packet data from the given packet and saves them.

- byte[] GetBytes ()

  The function returns the data header of the currently saved sequence.

Public Attributes

- const int HeaderLengthInOctets = 41

  The specified length of the packet header in octets.

- const byte PacketId = 0

  The specified id of the packet type. It is used to identify a packet.

Properties

- string SourceGuid `[get]`

  The function to get the Guid identifying the sender.

- Int64 Sequence `[get, set]`

  The function to get and set the sequence number of this packet.

- Int32 TotalPacketCount `[get]`

  The function to get the total count of packets in the related transfer sequence.

- Int32 TotalPayloadDataLength `[get]`

  The function to get the total length of payload data in the related transfer sequence.

- Int32 OriginatingDataLength `[get]`

  The function to get the original length of the data prior to encoding it into the transfer format.

- Int16 MediaFormat `[get]`

  The function to get the format of media in the packet.

- Int16 MediaTransferCompression `[get]`

  The function to get the compression format of media in the packet.

- byte[] PayloadData `[get]`

  The function to get the payload data in this packet.

2.5.1 Detailed Description

The class represents media information and payload data up to one network MTU in length from a MediaPacket. MediaHeaderPackets are used to transfer a portion of payload data and the media information from a MediaPacket. The combination of a MediaHeaderPacket and the subsequent MediaContinuationPackets is a transfer sequence.

<author>Veli-Mikko Puupponen</author> The packet consist of a packet type identifier, sender GUID, a Int64 sequence number, media type identifier, media compression identified, payload data length and originating data length.

The packet must always have payload data. If the payload data length is smaller than network MTU, the transfer sequence can consist of only a single MediaHeaderPacket.

2.5.2 Constructor & Destructor Documentation

2.5.2.1 MediaHeaderPacket ( )

The function initializes a new MediaHeaderPacket instance with no parameters.

2.5.2.2 MediaHeaderPacket (byte[] *guid,* Int16 *mediaTypeIdentifier,* Int16 *mediaTransferCompressioIdentifier,* Int32 *packetCount,* Int32 *totalPayloadDataLength,* Int32 *originatingDataLength,* byte[] *payload,* Int64 *sequenceNumber*)

The function initializes a new MediaHeaderPacket instance that has the specified source Guid, media type identified, media transfer compression identified, packet count, total payload data length, originating data length, sequence number and payload data

Parameters

guid	Sthe string representation of the sender's Guid.
mediaType↩ Identifier	The payload media type.
mediaTransfer↩ Compressio↩ Identifier	The payload media transfer compression type.
packetCount	Total count of packets in the related trasfer sequence.
totalPayload↩ DataLength	The total number of payload octets in the complete related trasfer sequence.
originating↩ DataLength	The original length of the data before encoding into the transfer format.
payload	The payload data in this packet.
sequence↩ Number	The squence number for this packet, must not vary within a transfer sequence.

2.5.2.3 MediaHeaderPacket (byte[] *guid,* Int16 *mediaTypeIdentifier,* Int16 *mediaTransferCompressioIdentifier,* Int32 *packetCount,* Int32 *totalPayloadDataLength,* Int32 *originatingDataLength,* byte[] *payload*)

The function initializes a new MediaHeaderPacket instance that has the specified source Guid, media type identified, media transfer compression identified, packet count, total payload data length, originating data length and payload data

The sequence number must be provided prior any calls to the parameterless GetBytes.

Parameters

guid	Sthe string representation of the sender's Guid.
mediaType↩ Identifier	The payload media type.
mediaTransfer↩ Compressio↩ Identifier	The payload media transfer compression type.
packetCount	Total count of packets in the related trasfer sequence.
totalPayload↩ DataLength	The total number of payload octets in the complete related trasfer sequence.
originating↩ DataLength	The original length of the data before encoding into the transfer format.
payload	The payload data in this packet.

2.5.3 Member Function Documentation

2.5.3.1 void FromBytes (byte[] *packetBytes*)

The function gets the media data, the packet control data and the packet data from the given packet and saves them.

Parameters

packetBytes	The given data.

Implements INetworkPacket.

2.5.3.2 byte [] GetBytes (long *sequenceNumber*)

The function saves the given sequence number and returns the data header of the given sequence.

Parameters

sequence↩ Number	The sequence to retrieve.

Returns

The data header of given the sequence.

2.5.3.3 byte [] GetBytes ( )

The function returns the data header of the currently saved sequence.

Returns

The data header of current the sequence.

Implements INetworkPacket.

2.5.4 Member Data Documentation

2.5.4.1 const int HeaderLengthInOctets = 41

The specified length of the packet header in octets.

2.5.4.2 const byte PacketId = 0

The specified id of the packet type. It is used to identify a packet.

2.5.5 Property Documentation

2.5.5.1 Int16 MediaFormat [get]

The function to get the format of media in the packet.

2.5.5.2 Int16 MediaTransferCompression [get]

The function to get the compression format of media in the packet.

2.5.5.3 Int32 OriginatingDataLength [get]

The function to get the original length of the data prior to encoding it into the transfer format.

2.5.5.4 byte [] PayloadData [get]

The function to get the payload data in this packet.

2.5.5.5 Int64 Sequence [get], [set]

The function to get and set the sequence number of this packet.

2.5.5.6 string SourceGuid [get]

The function to get the Guid identifying the sender.

2.5.5.7 Int32 TotalPacketCount [get]

The function to get the total count of packets in the related transfer sequence.

2.5.5.8 Int32 TotalPayloadDataLength [get]

The function to get the total length of payload data in the related transfer sequence.

The documentation for this class was generated from the following file:

- MediaHeaderPacket.cs

2.6 MediaInformation Class Reference

The class for media type information. It is used to the format and the compression of a media.

Public Member Functions

- MediaInformation (MediaFormat format, TransferCompression compression)

  The function initializes a new MediaFormat instance with the specified media type parameters.

Properties

- MediaFormat Format [get, set]

  The function to set and get the media format.
- TransferCompression Compression [get, set]

  The function to set and get the media compression format.

2.6.1 Detailed Description

The class for media type information. It is used to the format and the compression of a media.

<author>Veli-Mikko Puupponen</author>

2.6.2 Constructor & Destructor Documentation

2.6.2.1 MediaInformation (MediaFormat *format,* TransferCompression *compression*)

The function initializes a new MediaFormat instance with the specified media type parameters.

Parameters

format	The format of the media.
compression	The media transfer compression format.

2.6.3 Property Documentation

2.6.3.1 TransferCompression Compression [get],[set]

The function to set and get the media compression format.

2.6.3.2 MediaFormat Format [get],[set]

The function to set and get the media format.

The documentation for this class was generated from the following file:

- MediaInformation.cs

2.7 MediaPacket Class Reference

The class representts a user-level media packet. The packet is used when receiving and sending media using the UdpMediaClientSocket. It encapsulates the complete media data and media type information. It can provide a collection of NetworkPackets conforming to the network MTU for network transfer. It can later be reconstructed from such packets at the receiving end.

Public Member Functions

- MediaPacket (MediaInformation mediaInfo, byte[] mediaPayload, Int64 sequenceNumber, string guid)

  The function nstantializes a new MediaPacket with the specified media information, payload data, sender Guid and sequence number.
- MediaPacket (MediaInformation mediaInfo, byte[] mediaPayload, string guid, Int32 originatingLength, Int64 sequenceNumber)

  The function nstantializes a new MediaPacket with the specified media information, payload data, sender Guid, original data length and sequence number.
- MediaPacket (MediaInformation mediaInfo, byte[] mediaPayload, string guid, Int32 originatingLength)

  The function nstantializes a new MediaPacket with the specified media information, payload data, sender Guid and original data length.
- MediaPacket (MediaInformation mediaInfo, byte[] mediaPayload, string guid)

  The function nstantializes a new MediaPacket with the specified media information, payload data and sender Guid.
- MediaPacket (MediaHeaderPacket newHeaderPacket)

  The function nstantializes a new MediaPacket using the MediaHeaderPacket from a network transfer packet sequence. If MediaPacket from which the MediaHeaderPacket was derived and had no more than one network MTU of payload data, this MediaPacket instance is now complete (HasAllContinuationPackets = true).
- MediaPacket (MediaContinuationPacket oneContinuationPacket)

  The function nstantializes a new MediaPacket using any MediaContinuationPacket from a network transfer packet sequence. All MediaPackets require at least the MediaHeaderPacket from the relevant packet sequence generated from the original MediaPacket. Therefore this MediaPacket will not be complete until at least the MediaHeaderPacket has been added using the method AddHeaderPacket.
- bool AddHeaderPacket (MediaHeaderPacket newHeaderPacket)

  The function adds a header packet to this MediaPacket. If there already is a header packet in this instance, it returns the HasAllContinuationPackets indicating whether some continuation packets are missing or not.
- bool AddContinuationPacket (MediaContinuationPacket continuationPacket)

  The function adds a MediaContinuationPacket to this MediaPacket. It returns a boolean indicaing if all Media↩ ContinuationPackets necessary to complete this MediaPacket have now been added.
- INetworkPacket[] GetAllTransmissionPackets (int mtu)

  The function converts this MediaPacket into a collection of NetworkPackets for network transfer. All resulting networkPackets will have a byte representation with length no more that the specified MTU.

Properties

- Int32 OriginatingDataLength [get]

  The function to get the original length of the data prior to encoding into the specified transfer format.
- Int64 Sequence [get, set]

  The function to get and set the sequence number of this packet.

- bool HasAllContinuationPackets [get]

  The function returns True, if not reconstructed from a network packet sequence or hass all packets from the relevant sequence. False otherwise.
- bool PayloadLengthMismatched [get]

  The function returns True, if the packet is reconstructed from a network packet sequence and one or more subpackets had an unexpected payload data length. Otherwise false.
- string SenderGuid [get]

  The function to get the string Guid identifying the sender.
- DateTime InitializationTime [get]

  The function to get the time of the instantialization of the instance.
- byte[] CompletePayloadData [get]

  The function to get the complete payload data for this instance.
- MediaInformation PacketMediaInformation [get]

  The function to get the MediaInformation describing the media in this instance. Can be null on a packet constructed from a subpacket transfer sequence.

2.7.1 Detailed Description

The class representts a user-level media packet. The packet is used when receiving and sending media using the UdpMediaClientSocket. It encapsulates the complete media data and media type information. It can provide a collection of NetworkPackets conforming to the network MTU for network transfer. It can later be reconstructed from such packets at the receiving end.

<author>Veli-Mikko Puupponen</author>

2.7.2 Constructor & Destructor Documentation

2.7.2.1 MediaPacket (MediaInformation *mediaInfo,* byte[] *mediaPayload,* Int64 *sequenceNumber,* string *guid*)

The function nstantializes a new MediaPacket with the specified media information, payload data, sender Guid and sequence number.

It throws an ArgumentException if any of the arguments is null or the payload data length is zero.

Parameters

mediaInfo	The MediaInformation instance describing the data contained in this instance.
mediaPayload	The media data.
sequence↩ Number	The sequence number for all subpackets generated from this packet during network transmission.
guid	The string representation of the sender's Guid.

2.7.2.2 MediaPacket (MediaInformation *mediaInfo,* byte[] *mediaPayload,* string *guid,* Int32 *originatingLength,* Int64 *sequenceNumber*)

The function nstantializes a new MediaPacket with the specified media information, payload data, sender Guid, original data length and sequence number.

It throws an ArgumentException if any of the arguments is null or the payload data length is zero.

Parameters

mediaInfo	The MediaInformation instance describing the data contained in this instance.
mediaPayload	The media data.

guid	The string representation of the sender's Guid.
originating↩ Length	The riginal length of the data prior to encoding into the specified transfer format.
sequence↩ Number	The sequence number for all subpackets generated from this packet during network transmission.

2.7.2.3 MediaPacket (MediaInformation *mediaInfo,* byte[] *mediaPayload,* string *guid,* Int32 *originatingLength*)

The function nstantializes a new MediaPacket with the specified media information, payload data, sender Guid and original data length.

It throws an ArgumentException if any of the arguments is null or the payload data length is zero.

Parameters

mediaInfo	The MediaInformation instance describing the data contained in this instance.
mediaPayload	The media data.
guid	The string representation of the sender's Guid.
originating↩ Length	The riginal length of the data prior to encoding into the specified transfer format.

2.7.2.4 MediaPacket (MediaInformation *mediaInfo,* byte[] *mediaPayload,* string *guid*)

The function nstantializes a new MediaPacket with the specified media information, payload data and sender Guid.

It throws an ArgumentException if any of the arguments is null, the payload data length is zero or the Guid format is incorrect.

Parameters

mediaInfo	The MediaInformation instance describing the data contained in this instance.
mediaPayload	The media data.
guid	The string representation of the sender's Guid.

2.7.2.5 MediaPacket (MediaHeaderPacket *newHeaderPacket*)

The function nstantializes a new MediaPacket using the MediaHeaderPacket from a network transfer packet sequence. If MediaPacket from which the MediaHeaderPacket was derived and had no more than one network MTU of payload data, this MediaPacket instance is now complete (HasAllContinuationPackets = true).

Otherwise the MediaContinuationPackets from the transfer sequence need to be added to this instace using the method AddContinuationPacket.

It throws an ArgumentException if the MediaHeaderPacket is not valid.

Parameters

newHeader↩ Packet	The MediaHeaderPacket instance.

2.7.2.6 MediaPacket (MediaContinuationPacket *oneContinuationPacket*)

The function nstantializes a new MediaPacket using any MediaContinuationPacket from a network transfer packet sequence. All MediaPackets require at least the MediaHeaderPacket from the relevant packet sequence generated from the original MediaPacket. Therefore this MediaPacket will not be complete until at least the MediaHeader↩ Packet has been added using the method AddHeaderPacket.

Otherwise the MediaContinuationPackets from the transfer sequence need to be added to this instace using the method AddContinuationPacket.

It throws an ArgumentException if the MediaHeaderPacket is not valid.

Parameters

continuation↩ Packet	The MediaContinuationPacket that is a part of the related transfer packet sequence.

2.7.3 Member Function Documentation

2.7.3.1 bool AddContinuationPacket (MediaContinuationPacket *continuationPacket*)

The function adds a MediaContinuationPacket to this MediaPacket. It returns a boolean indicaing if all Media↩ ContinuationPackets necessary to complete this MediaPacket have now been added.

It throws an ArgumentException if the MediaContinuationPacket is not valid or has a mismatching sequence number or guid.

Parameters

continuation↩ Packet	The MediaContinuationPacket that is a part of the related transfer packet sequence.

Returns

True, if this MediaPacket now has all continuation packets, otherwise false.

2.7.3.2 bool AddHeaderPacket (MediaHeaderPacket *newHeaderPacket*)

The function adds a header packet to this MediaPacket. If there already is a header packet in this instance, it returns the HasAllContinuationPackets indicating whether some continuation packets are missing or not.

If the MediaHeaderPacket is valid for this instance and no MediaContinuationPackets are missing, it returns true indicating that this instance is now complete.

It throws an ArgumentException if the MediaHeaderPacket is not valid or has a mismatching sequence number or guid.

Parameters

newHeader↩ Packet	The header packet for supplying first part of the payload data and overall information to this instance.

Returns

True, if this instance now has all subpackets from the relevant transfer packet sequence, otherwise false.

2.7.3.3 INetworkPacket [ ] GetAllTransmissionPackets (int *mtu*)

The function converts this MediaPacket into a collection of NetworkPackets for network transfer. All resulting networkPackets will have a byte representation with length no more that the specified MTU.

The MTU has to be 100 bytes or more. It throws an ArgumentException if MTU $<$ 100.

Parameters

mtu	The maximal network transmission unit size used to limit the resulting NetworkPacket payload sizes.

Returns

A collection of mediapackets from the given mediapacket.

2.7.4 Property Documentation

2.7.4.1 byte [] CompletePayloadData [get]

The function to get the complete payload data for this instance.

2.7.4.2 bool HasAllContinuationPackets [get]

The function returns True, if not reconstructed from a network packet sequence or hass all packets from the relevant sequence. False otherwise.

2.7.4.3 DateTime InitializationTime [get]

The function to get the time of the instantialization of the instance.

2.7.4.4 Int32 OriginatingDataLength [get]

The function to get the original length of the data prior to encoding into the specified transfer format.

2.7.4.5 MediaInformation PacketMediaInformation [get]

The function to get the MediaInformation describing the media in this instance. Can be null on a packet constructed from a subpacket transfer sequence.

2.7.4.6 bool PayloadLengthMismatched [get]

The function returns True, if the packet is reconstructed from a network packet sequence and one or more subpackets had an unexpected payload data length. Otherwise false.

2.7.4.7 string SenderGuid [get]

The function to get the string Guid identifying the sender.

2.7.4.8 Int64 Sequence [get], [set]

The function to get and set the sequence number of this packet.

The documentation for this class was generated from the following file:

- MediaPacket.cs

2.8 PingPacket Class Reference

The class represents a ping packet for the protocol defined and implemented by the UdpMediaClientSocket. The packet consist of a packet type identifier, a sender GUID and a Int64 sequence number.

Inherits INetworkPacket.

Public Member Functions

- PingPacket ()

  The function to initializes a new ping packet with no information.
- PingPacket (string guid, Int64 sequenceNumber)

  The function initializes a new ping packet that has the provided sequence number and sender GUID.
- PingPacket (string guid)

  The function initializes a new ping packet that has the provided sender GUID.
- byte[] GetBytes (long sequenceNumber)

  The function saves the given sequence number and reurns the data of the given sequence.
- void FromBytes (byte[] packetBytes)

  The function gets the guid and the sequence from the data header of the given data and saves them.

- byte[] GetBytes ()

  The function returns the data of the given sequence.

Public Attributes

- const int HeaderLengthInOctets = 25

  The specified length of the packet header in octects.
- const byte PacketId = 3

  The specified id of the packet type. It is used to identify a packet.

Properties

- string SourceGuid `[get]`

  String represenstation of the Guid identifying the sender.
- Int64 Sequence `[get, set]`

  The function sets and gets the packet sequence number.

2.8.1 Detailed Description

The class represents a ping packet for the protocol defined and implemented by the UdpMediaClientSocket. The packet consist of a packet type identifier, a sender GUID and a Int64 sequence number.

<author>Veli-Mikko Puupponen</author>

2.8.2 Constructor & Destructor Documentation

2.8.2.1 PingPacket ( )

The function to initializes a new ping packet with no information.

2.8.2.2 PingPacket (string *guid*, Int64 *sequenceNumber*)

The function initializes a new ping packet that has the provided sequence number and sender GUID.

Parameters

guid	Guid of the sender.
sequence↩ Number	Packet sequence number.

2.8.2.3 PingPacket (string *guid*)

The function initializes a new ping packet that has the provided sender GUID.

Parameters

guid	The guid of the sender.

2.8.3 Member Function Documentation

2.8.3.1 void FromBytes (byte[] *packetBytes*)

The function gets the guid and the sequence from the data header of the given data and saves them.

Parameters

packetBytes	The given data.

Implements INetworkPacket.

2.8.3.2 byte [] GetBytes (long *sequenceNumber*)

The function saves the given sequence number and reurns the data of the given sequence.

Parameters

sequence↩ Number	The sequence to retrive.

Returns

The data of given the sequence.

2.8.3.3 byte [] GetBytes ( )

The function returns the data of the given sequence.

Returns

The packet data.

Implements INetworkPacket.

2.8.4 Member Data Documentation

2.8.4.1 const int HeaderLengthInOctets = 25

The specified length of the packet header in octects.

2.8.4.2 const byte PacketId = 3

The specified id of the packet type. It is used to identify a packet.

2.8.5 Property Documentation

2.8.5.1 Int64 Sequence [get],[set]

The function sets and gets the packet sequence number.

2.8.5.2 string SourceGuid [get]

String represenstation of the Guid identifying the sender.

The documentation for this class was generated from the following file:

- PingPacket.cs

2.9 UdpMediaClientSocket Class Reference

The class is client socket implementing a simple UDP-based media transfer protocol. Delivery of packets is not guaranteed. Sent packets are split to conform to the specified MTU. The underlaying IP checksum is relied upon for the integrity of the packets. The constant ping packets are employed to monitor the connection and to keep the UDP packets routed in both directions by the network channel.

Public Member Functions

- UdpMediaClientSocket (string serverIpAddress, int serverPort, int localPort, string clientGuid, int networkMtu, int timeOutMillis)

  The function initializes a new UdpMediaClientSocket to connect to the specified host. The provided GUID is used for identifying this client during the connection. The socket is not connected before a call to the Connect method. The socket will use the specified MTU on the network packets. The underlying UDP socket will use the specified port at the local system.
- UdpMediaClientSocket (string serverIpAddress, int serverPort, int localPort, string clientGuid, int networkMtu)

  The function initializes a new UdpMediaClientSocket to connect to the specified host. The provided GUID is used for identifying this client during the connection. The socket is not connected before a call to the Connect method. The socket will use the specified MTU on the network packets. The underlying UDP socket will use the specified port at the local system.
- UdpMediaClientSocket (string serverIpAddress, int serverPort, string clientGuid, int networkMtu)

  The function initializes a new UdpMediaClientSocket to connect to the specified host. The provided GUID is used for identifying this client during the connection. The socket is not connected before a call to the Connect method. The socket will use the specified MTU on the network packets.
- UdpMediaClientSocket (string serverIpAddress, int serverPort, string clientGuid)

  The function initializes a new UdpMediaClientSocket to connect to the specified host. The provided GUID is used for identifying this client during the connection. The socket is not connected before a call to the Connect method. The socket will use a default MTU of 1024 bytes on the network.
- UdpMediaClientSocket (int pingLossLimit, string serverIpAddress, int serverPort, string clientGuid)

  The function initializes a new UdpMediaClientSocket to connect to the specified host. The provided GUID is used for identifying this client during the connection. The socket is not connected before a call to the Connect method. The socket will use a default MTU of 1024 bytes on the network.
- void Disconnect ()

  The function dsconnets the connected instance.
- void Connect ()

  The function connects a non-connected instance. This method needs to be called prior to any send operations for them to succeed.
- bool SendMedia (MediaInformation mediaInfo, byte[] mediaData, int mediaOriginalLength)

  The function sends a media to the remote host. Retuns true if the packet was successfully enqueued to the outgoing media packet queue. If the socket is not connected, has become faulted or the outgoing packet queue is already filled, returns false.
- bool SendRoutingRequest (string fromGuid, string toGuid, bool enableRouting)

  The function sends a routing request to the remote host. The socket enforces no rules on the guids that are specified on the request.

Public Attributes

- MediaPacketReceived MediaPacketReceivedEvent

  The event for the MediaPacketReceived delegate function.
- ControlPacketReceived ControlPacketReceivedEvent

  The event for the ControlPacketReceived delegate function.
- SuspectedConnectionProblem SuspectedConnectionProblemEvent

  The event for the SuspectedConnectionProblem delegate function.
- ConnectionFailed ConnectionFailedEvent

  The event for the ConnectionFailed delegate function.

Properties

- int SocketTimeoutMilliseconds `[get, set]`

  The function to set and get the socket timeout in milliseconds.
- bool IsFaulted `[get]`

The function returns True, if the socket has been enabled and the connection to the remote host has failed. Such a socket is no longer enabled. Otherwise false.

- bool IsEnabled [get]

  The function returns true, if the socket is enabled and functional. Otherwise false.

2.9.1 Detailed Description

The class is client socket implementing a simple UDP-based media transfer protocol. Delivery of packets is not guaranteed. Sent packets are split to conform to the specified MTU. The underlaying IP checksum is relied upon for the integrity of the packets. The constant ping packets are employed to monitor the connection and to keep the UDP packets routed in both directions by the network channel.

<author>Veli-Mikko Puupponen</author> This socket is usable in both peer-to-peer and server connection scenarios.

The existance of a remote host is monitored with constant ping packets. If excessive number of consecutive ping replies is lost, the socket is deemed faulted and the ConnectionFailedEvent is fired.

Uses separate sequence number counters for control, ping and media transfer packets.

2.9.2 Constructor & Destructor Documentation

2.9.2.1 UdpMediaClientSocket (string *serverIpAddress,* int *serverPort,* int *localPort,* string *clientGuid,* int *networkMtu,* int *timeOutMillis*)

The function initializes a new UdpMediaClientSocket to connect to the specified host. The provided GUID is used for identifying this client during the connection. The socket is not connected before a call to the Connect method. The socket will use the specified MTU on the network packets. The underlying UDP socket will use the specified port at the local system.

Parameters

serverIpAddress	The ip address of the server or peer in the standard string format.
serverPort	The port at the remote host.
localPort	The port at the local system.
clientGuid	The guid identifying this client.
networkMtu	The MTU for the underlying network.
timeOutMillis	The time-out limit for send operations on the underlying UDP socket.

2.9.2.2 UdpMediaClientSocket (string *serverIpAddress,* int *serverPort,* int *localPort,* string *clientGuid,* int *networkMtu*)

The function initializes a new UdpMediaClientSocket to connect to the specified host. The provided GUID is used for identifying this client during the connection. The socket is not connected before a call to the Connect method. The socket will use the specified MTU on the network packets. The underlying UDP socket will use the specified port at the local system.

Parameters

serverIpAddress	The ip address of the server or peer in the standard string format.
serverPort	The port at the remote host.
localPort	the port at the local system.
clientGuid	The guid identifying this client.
networkMtu	The MTU for the underlying network.

2.9.2.3 UdpMediaClientSocket (string *serverIpAddress,* int *serverPort,* string *clientGuid,* int *networkMtu*)

The function initializes a new UdpMediaClientSocket to connect to the specified host. The provided GUID is used for identifying this client during the connection. The socket is not connected before a call to the Connect method. The socket will use the specified MTU on the network packets.

Parameters

serverIpAddress	The ip address of the server or peer in the standard string format.
serverPort	The port at the remote host.
clientGuid	The guid identifying this client.
networkMtu	The MTU for the underlying network.

2.9.2.4 UdpMediaClientSocket (string *serverIpAddress,* int *serverPort,* string *clientGuid*)

The function initializes a new UdpMediaClientSocket to connect to the specified host. The provided GUID is used for identifying this client during the connection. The socket is not connected before a call to the Connect method. The socket will use a default MTU of 1024 bytes on the network.

Parameters

serverIpAddress	The ip address of the server or peer in the standard string format.
serverPort	The port at the remote host.
clientGuid	The guid identifying this client.

2.9.2.5 UdpMediaClientSocket (int *pingLossLimit,* string *serverIpAddress,* int *serverPort,* string *clientGuid*)

The function initializes a new UdpMediaClientSocket to connect to the specified host. The provided GUID is used for identifying this client during the connection. The socket is not connected before a call to the Connect method. The socket will use a default MTU of 1024 bytes on the network.

Parameters

pingLossLimit	The meximum allowable number of lost consecutive ping packets.
serverIpAddress	The ip address of the server or peer in the standard string format.
serverPort	The port at the remote host.
clientGuid	The guid identifying this client.

2.9.3 Member Function Documentation

2.9.3.1 void Connect ( )

The function connects a non-connected instance. This method needs to be called prior to any send operations for them to succeed.

2.9.3.2 void Disconnect ( )

The function dsconnets the connected instance.

2.9.3.3 bool SendMedia (MediaInformation *mediaInfo,* byte[] *mediaData,* int *mediaOriginalLength*)

The function sends a media to the remote host. Retuns true if the packet was successfully enqueued to the outgoing media packet queue. If the socket is not connected, has become faulted or the outgoing packet queue is already filled, returns false.

Parameters

mediaInfo	The information describing the outgoing media.
mediaData	The media data bytes.
mediaOriginal↩ Length	The length of the original data in case of compressions that require this value for decompression at the remote end.

Returns

True if the packet was successfully enqueued to the outgoing media packet queue, otherwise False.

2.9.3.4 bool SendRoutingRequest (string *fromGuid,* string *toGuid,* bool *enableRouting*)

The function sends a routing request to the remote host. The socket enforces no rules on the guids that are specified on the request.

Parameters

fromGuid	The guid of the source for the routing request.
toGuid	The guid of the target for the routing request.
enableRouting	Whether routing between the client sockets using the specified should be enabled or disabled.

Returns

True if the routing request was successfully sent, otherwise False.

2.9.4 Member Data Documentation

2.9.4.1 ConnectionFailed ConnectionFailedEvent

The event for the ConnectionFailed delegate function.

2.9.4.2 ControlPacketReceived ControlPacketReceivedEvent

The event for the ControlPacketReceived delegate function.

2.9.4.3 MediaPacketReceived MediaPacketReceivedEvent

The event for the MediaPacketReceived delegate function.

2.9.4.4 SuspectedConnectionProblem SuspectedConnectionProblemEvent

The event for the SuspectedConnectionProblem delegate function.

2.9.5 Property Documentation

2.9.5.1 bool IsEnabled [get]

The function returns true, if the socket is enabled and functional. Otherwise false.

2.9.5.2 bool IsFaulted [get]

The function returns True, if the socket has been enabled and the connection to the remote host has failed. Such a socket is no longer enabled. Otherwise false.

2.9.5.3 int SocketTimeoutMilliseconds [get],[set]

The function to set and get the socket timeout in milliseconds.

The documentation for this class was generated from the following file:

- UdpMediaClientSocket.cs