RTP Audio System

2.0.0

Generated by Doxygen 1.7.6.1

Sun Aug 19 2012 09:59:07

Contents

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 2

Class Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

Namespace Documentation

5.1 RTPConstants Namespace Reference

Variables

- const cardinal RTPMaxPayloadLimit = 8192
- const cardinal RTPDefaultMaxPayload = 1376
- const cardinal RTPDefaultHeaderSize = 12
- const card8 RTPVersion = 2
- const double RTPMicroSecondsPerTimeStamp = 1000.0 / 16.0
- const cardinal RTPMaxQualityLayers = 16

5.1.1 Variable Documentation

5.1.1.1 const cardinal RTPConstants::RTPDefaultHeaderSize = 12

Default RTP header size (CC = 0).

5.1.1.2 const cardinal RTPConstants::RTPDefaultMaxPayload = 1376

RTP default maximum payload is 1376 = 1500 - (12 + 16 ∗ 4) - 40 - 8 = Maximum ethernet data length - RTPPacket size - (16 ∗ CSRC) - UDP header size - IPv6 header size.

5.1.1.3 const cardinal RTPConstants::RTPMaxPayloadLimit = 8192

RTP maximum payload limit.

5.1.1.4 const cardinal RTPConstants::RTPMaxQualityLayers = 16

Maximum number of layers in one stream. Note: This is ∗not∗ a constant of RFC 1889 but a limit for the RTP structs!

5.1.1.5 const double RTPConstants::RTPMicroSecondsPerTimeStamp = 1000.0 / 16.0

Constant for microseconds per RTP timestamp.

5.1.1.6 const card8 RTPConstants::RTPVersion = 2

Constant for RTP version.

Chapter 6

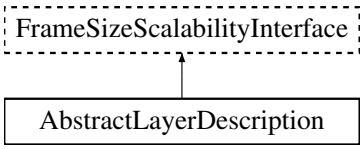
Class Documentation

6.1 AbstractLayerDescription Class Reference

Abstract Layer Description.

`#include <abstractlayerdescription.h>`

Inheritance diagram for AbstractLayerDescription:



Public Types

- enum LayerFlags { LF_BaseLayer = 0, LF_ExtensionLayer = (1 << 0) }

Public Member Functions

- AbstractLayerDescription ()
- virtual ~AbstractLayerDescription ()
- void initLayer (const cardinal pktHeaderSize, const cardinal pktMaxSize, const double maxTransferDelay, const cardinal maxBufferDelay, const double maxLossRate, const double maxJitter, const cardinal flags)
- card64 getBandwidth () const
- bool setBandwidth (const double frameRate, const card64 bandwidth)
- virtual cardinal getPacketRate (const double frameRate) const
- card64 bandwidthToBandwidth (const card64 bandwidth, const double frameRate, const cardinal bufferDelay, const cardinal newBufferDelay) const

- card64 payloadBandwidthToBandwidth (const card64 bandwidth, const double frameRate, const cardinal bufferDelay, const cardinal newBufferDelay) const
- virtual cardinal frameSizeToPacketRate (const double frameRate, const cardinal frameSize) const
- double getMaxTransferDelay () const
- void setMaxTransferDelay (const double maxDelay)
- double getMaxLossRate () const
- void setMaxLossRate (const double maxLossRate)
- double getMaxJitter () const
- void setMaxJitter (const double maxJitter)
- bool isValidFrameSize (const double frameRate, const cardinal bufferDelay, const cardinal size) const
- cardinal getNearestValidFrameSize (const double frameRate, const cardinal bufferDelay, const cardinal size) const
- virtual cardinal payloadToRaw (const double frameRate, const cardinal payload, const cardinal bufferDelay) const
- virtual cardinal rawToPayload (const double frameRate, const cardinal raw, const cardinal bufferDelay) const
- cardinal getMinFrameSize (const double frameRate) const
- cardinal getMaxFrameSize (const double frameRate) const
- cardinal getPeakFrameSizeForSize (const double frameRate, const cardinal frameSize) const
- cardinal getPacketCountForSize (const double frameRate, const cardinal frameSize) const
- double getPrevFrameSizeForSize (const double frameRate, const cardinal frameSize) const
- double getNextFrameSizeForSize (const double frameRate, const cardinal frameSize) const
- double getFrameSizeScaleFactorForSize (const double frameRate, const cardinal frameSize) const
- double getFrameSizeUtilizationForSize (const double frameRate, const cardinal frameSize) const
- cardinal getMinFrameSizeForDelay (const double frameRate, const cardinal bufferDelay) const
- cardinal getMaxFrameSizeForDelay (const double frameRate, const cardinal bufferDelay) const
- cardinal getPeakFrameSizeForDelayAndSize (const double frameRate, const cardinal bufferDelay, const cardinal frameSize) const
- cardinal getPacketCountForDelayAndSize (const double frameRate, const cardinal bufferDelay, const cardinal frameSize) const
- double getPrevFrameSizeForDelayAndSize (const double frameRate, const cardinal bufferDelay, const cardinal frameSize) const
- double getNextFrameSizeForDelayAndSize (const double frameRate, const cardinal bufferDelay, const cardinal frameSize) const
- double getFrameSizeScaleFactorForDelayAndSize (const double frameRate, const cardinal bufferDelay, const cardinal frameSize) const
- double getFrameSizeUtilizationForDelayAndSize (const double frameRate, const cardinal bufferDelay, const cardinal frameSize) const

- cardinal getBufferDelay () const
- cardinal setBufferDelay (const cardinal bufferDelay)
- cardinal getPrevBufferDelay (const double frameRate) const
- cardinal getNextBufferDelay (const double frameRate) const
- InternetAddress getSource () const
- InternetFlow getDestination () const
- void setSource (const InternetAddress &source)
- void setDestination (const InternetFlow &destination)
- cardinal getFlags () const
- void setFlags (const cardinal flags)

Static Public Member Functions

- static card64 frameSizeToBandwidth (const double frameRate, const cardinal frameSize)
- static cardinal bandwidthToFrameSize (const double frameRate, const card64 bandwidth)

Protected Attributes

- cardinal PktHeaderSize
- cardinal PktMaxSize
- card64 Bandwidth
- double MaxTransferDelay
- double MaxLossRate
- double MaxJitter
- cardinal BufferDelay
- cardinal MaxBufferDelay
- cardinal Flags
- InternetAddress Source
- InternetFlow Destination

6.1.1 Detailed Description

Abstract Layer Description.

This class contains a layer's QoS requirements. Important node: All frames sizes in this class are ∗raw∗ frame sizes, the frames sizes in FrameSizeScalability are payload frame sizes. This class does necessary translation.

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

6.1.2 Member Enumeration Documentation

6.1.2.1 enum AbstractLayerDescription::LayerFlags

Layer flags.

Enumerator:

LF_BaseLayer

LF_ExtensionLayer

6.1.3 Constructor & Destructor Documentation

6.1.3.1 AbstractLayerDescription::AbstractLayerDescription ( )

Constructor.

6.1.3.2 AbstractLayerDescription::~AbstractLayerDescription ( ) [virtual]

Destructor.

6.1.4 Member Function Documentation

6.1.4.1 card64 AbstractLayerDescription::bandwidthToBandwidth (const card64 *bandwidth*, const double *frameRate*, const cardinal *bufferDelay*, const cardinal *newBufferDelay*) const [inline]

Translate bandwidth into bandwidth using different buffer delay.

Parameters

bandwidth	Input bandwidth.
frameRate	Input frame rate.
bufferDelay	Input buffer delay.
newBuffer-Delay	Output buffer delay.

Returns

Output bandwidth.

6.1.4.2 static cardinal AbstractLayerDescription::bandwidthToFrameSize (const double *frameRate*, const card64 *bandwidth*) [inline, static]

Translate bandwidth into frame size.

Parameters

frameRate	Frame rate.
bandwidth	Bandwidth.

Returns

Frame size.

6.1.4.3 static card64 AbstractLayerDescription::frameSizeToBandwidth (const double *frameRate*, const cardinal *frameSize*) [inline, static]

Translate frame size into bandwidth.

Parameters

frameRate	Frame rate.
frameSize	Frame size.

Returns

Bandwidth.

6.1.4.4 cardinal AbstractLayerDescription::frameSizeToPacketRate (const double *frameRate*, const cardinal *frameSize*) const [virtual]

Get packets per second for given frame size.

Parameters

frameRate	Frame rate.
frameSize	Frame size.

Returns

Packets per second.

6.1.4.5 card64 AbstractLayerDescription::getBandwidth ( ) const [inline]

Get bandwidth.

Returns

Bandwidth.

6.1.4.6 cardinal AbstractLayerDescription::getBufferDelay ( ) const [inline]

Get buffer delay.

Returns

Buffer delay in frame rate units.

6.1.4.7 InternetFlow AbstractLayerDescription::getDestination ( ) const [inline]

Get destination address.

Returns

Destination address.

6.1.4.8 cardinal AbstractLayerDescription::getFlags ( ) const [inline]

Get flags.

6.1.4.9 double AbstractLayerDescription::getFrameSizeScaleFactorForDelayAndSize (const double *frameRate,* const cardinal *bufferDelay,* const cardinal *frameSize*) const [inline]

Get frame size scale factor for given frame rate, size and buffer delay. (size - MinFrameSize) / (MaxFrameSize - MinFrameSize).

Parameters

frameRate	Frame rate.
frameSize	Frame size.

Returns

Scale factor (out of [0,1]).

6.1.4.10 double AbstractLayerDescription::getFrameSizeScaleFactorForSize (const double *frameRate,* const cardinal *frameSize*) const [inline]

Get frame size scale factor for given frame rate and size: (size - MinFrameSize) / (MaxFrameSize - MinFrameSize).

Parameters

frameRate	Frame rate and size.
frameSize	Frame size.

Returns

Scale factor (out of [0,1]).

6.1.4.11 double AbstractLayerDescription::getFrameSizeUtilizationForDelayAndSize (const double *frameRate,* const cardinal *bufferDelay,* const cardinal *frameSize*) const [inline]

Get frame size utilization for given frame rate, size and buffer delay.

Parameters

frameRate	Frame rate.
frameSize	Frame size.

Returns

Utilization (out of [0,1]).

6.1.4.12 double AbstractLayerDescription::getFrameSizeUtilizationForSize (const double *frameRate,* const cardinal *frameSize*) const [inline]

Get frame size utilization for given frame rate and size.

Parameters

frameRate	Frame rate and size.
frameSize	Frame size.

Returns

Utilization (out of [0,1]).

6.1.4.13 cardinal AbstractLayerDescription::getMaxFrameSize (const double *frameRate*) const [inline]

Get maximum frame size for given frame rate.

Parameters

frameRate	Frame rate.

Returns

Maximum frame size.

6.1.4.14 cardinal AbstractLayerDescription::getMaxFrameSizeForDelay (const double *frameRate,* const cardinal *bufferDelay*) const [inline]

Get maximum frame size for given frame rate and buffer delay.

Parameters

frameRate	Frame rate.

Returns

Maximum frame size.

6.1.4.15 double AbstractLayerDescription::getMaxJitter ( ) const [inline]

Get maximum jitter.

Returns

Maximum jitter in microseconds.

6.1.4.16 double AbstractLayerDescription::getMaxLossRate ( ) const [inline]

Get maximum loss rate.

Returns

Maximum loss rate (out of [0,1]).

6.1.4.17 double AbstractLayerDescription::getMaxTransferDelay ( ) const [inline]

Get maximum transfer delay.

Returns

Maximum transfer delay in microseconds.

6.1.4.18 cardinal AbstractLayerDescription::getMinFrameSize (const double *frameRate*) const [inline]

Get minimum frame size for given frame rate.

Parameters

frameRate	Frame rate.

Returns

Minimum frame size.

6.1.4.19 cardinal AbstractLayerDescription::getMinFrameSizeForDelay (const double *frameRate,* const cardinal *bufferDelay*) const [inline]

Get minimum frame size for given frame rate and buffer delay.

Parameters

frameRate	Frame rate.

Returns

Minimum frame size.

6.1.4.20 cardinal AbstractLayerDescription::getNearestValidFrameSize (const double *frameRate,* const cardinal *bufferDelay,* const cardinal *size*) const [inline]

Get nearest lower frame size for given frame rate and buffer delay.

Parameters

frameRate	Frame rate.
bufferDelay	Buffer delay.
size	FrameSize.

Returns

Nearest lower frame size.

6.1.4.21 cardinal AbstractLayerDescription::getNextBufferDelay (const double *frameRate*) const [inline]

Get next higher buffer delay.

Parameters

frameRate	Frame rate.

Returns

Buffer delay in frame rate units.

6.1.4.22 double AbstractLayerDescription::getNextFrameSizeForDelayAndSize (const double *frameRate,* const cardinal *bufferDelay,* const cardinal *frameSize*) const [inline]

Get next higher frame size for given frame rate, size and buffer delay.

Parameters

frameRate	Frame rate.
frameSize	Frame size.

Returns

Next higher frame size.

6.1.4.23 double AbstractLayerDescription::getNextFrameSizeForSize (const double *frameRate,* const cardinal *frameSize*) const [inline]

Get next higher frame size for given frame rate and size.

Parameters

frameRate	Frame rate and size.
frameSize	Frame size.

Returns

Next higher frame size.

6.1.4.24 cardinal AbstractLayerDescription::getPacketCountForDelayAndSize (const double *frameRate,* const cardinal *bufferDelay,* const cardinal *frameSize*) const [inline]

Get number of packets (upper limit) for given frame rate, size and buffer delay.

Parameters

frameRate	Frame rate.

Returns

Number of packets.

6.1.4.25 cardinal AbstractLayerDescription::getPacketCountForSize (const double *frameRate,* const cardinal *frameSize*) const [inline]

Get number of packets (upper limit) for given frame rate and size.

Parameters

frameRate	Frame rate.

Returns

Number of packets.

6.1.4.26 cardinal AbstractLayerDescription::getPacketRate (const double *frameRate*) const [virtual]

Get packet rate.

Parameters

frameRate	Frame rate.

Returns

Bandwidth limit.

6.1.4.27 cardinal AbstractLayerDescription::getPeakFrameSizeForDelayAndSize (const double *frameRate,* const cardinal *bufferDelay,* const cardinal *frameSize*) const [inline]

Get peak frame size for given frame rate, size and buffer delay.

Parameters

frameRate	Frame rate.

Returns

Peak frame size.

6.1.4.28 cardinal AbstractLayerDescription::getPeakFrameSizeForSize (const double *frameRate,* const cardinal *frameSize*) const [inline]

Get peak frame size for given frame rate and size.

Parameters

frameRate	Frame rate.

Returns

Peak frame size.

6.1.4.29 cardinal AbstractLayerDescription::getPrevBufferDelay (const double *frameRate*) const [inline]

Get next lower buffer delay.

Parameters

frameRate	Frame rate.

Returns

Buffer delay in frame rate units.

6.1.4.30 double AbstractLayerDescription::getPrevFrameSizeForDelayAndSize (const double *frameRate,* const cardinal *bufferDelay,* const cardinal *frameSize*) const [inline]

Get next lower frame size for given frame rate, size and buffer delay.

Parameters

frameRate	Frame rate.
frameSize	Frame size.

Returns

Next lower frame size.

6.1.4.31 double AbstractLayerDescription::getPrevFrameSizeForSize (const double *frameRate,* const cardinal *frameSize*) const [inline]

Get next lower frame size for given frame rate and size and size.

Parameters

frameRate	Frame rate.
frameSize	Frame size.

Returns

Next lower frame size.

6.1.4.32 InternetAddress AbstractLayerDescription::getSource ( ) const [inline]

Get source address.

Returns

Source address.

6.1.4.33 void AbstractLayerDescription::initLayer (const cardinal *pktHeaderSize,* const cardinal *pktMaxSize,* const double *maxTransferDelay,* const cardinal *maxBufferDelay,* const double *maxLossRate,* const double *maxJitter,* const cardinal *flags*) [inline]

Initialize layer description.

Parameters

pktHeader-Size	Packet header size, e.g. 40 + 8 + 12 (IPv6 + UDP + RTP).
pktMaxSize	Maximum packet size, e.g. 1500.
max-Transfer-Delay	Maximum transfer delay in microseconds.
maxBuffer-Delay	Maximum buffer delay in frame rate units.
maxLoss-Rate	Maximum loss rate (out of [0,1]).
maxJitter	Maximum jitter in microseconds.

6.1.4.34 bool AbstractLayerDescription::isValidFrameSize (const double *frameRate,* const cardinal *bufferDelay,* const cardinal *size*) const [inline]

Check, if given frame size is valid for given frame rate and buffer delay.

Parameters

frameRate	Frame rate.
bufferDelay	Buffer delay.
size	FrameSize.

Returns

true, if frame size is valid; false otherwise.

6.1.4.35 card64 AbstractLayerDescription::payloadBandwidthToBandwidth (const card64 *bandwidth,* const double *frameRate,* const cardinal *bufferDelay,* const cardinal *newBufferDelay*) const

Translate ∗payload∗ bandwidth into bandwidth using different buffer delay.

Parameters

bandwidth	Input payload bandwidth.
frameRate	Input frame rate.
bufferDelay	Input buffer delay.
newBuffer-Delay	Output buffer delay.

Returns

Output payload bandwidth.

6.1.4.36 cardinal AbstractLayerDescription::payloadToRaw (const double *frameRate,* const cardinal *payload,* const cardinal *bufferDelay*) const [virtual]

Translate payload frame size into raw frame size.

Parameters

frameRate	Frame rate.
payload	Payload frame size.
bufferDelay	Buffer delay.

Returns

Raw frame size.

6.1.4.37 cardinal AbstractLayerDescription::rawToPayload (const double *frameRate,* const cardinal *raw,* const cardinal *bufferDelay*) const [virtual]

Translate raw frame size into payload frame size.

Parameters

frameRate	Frame rate.
raw	Raw frame size.
bufferDelay	Buffer delay.

Returns

Payload frame size.

6.1.4.38 bool AbstractLayerDescription::setBandwidth (const double *frameRate,* const card64 *bandwidth*) [inline]

Set bandwidth.

Parameters

frameRate	Frame rate.
bandwidth	Bandwidth.

Returns

true, if bandwidth is sufficient for minimum requirement.

6.1.4.39 cardinal AbstractLayerDescription::setBufferDelay (const cardinal *bufferDelay*)

Set buffer delay.

Parameters

bufferDelay	Buffer delay in frame rate units.

Returns

Buffer delay set in frame rate units.

6.1.4.40 void AbstractLayerDescription::setDestination (const InternetFlow & *destination*) [inline]

Set destination address.

Parameters

destination	Destination address

6.1.4.41 void AbstractLayerDescription::setFlags (const cardinal *flags*) [inline]

Set flags.

6.1.4.42 void AbstractLayerDescription::setMaxJitter (const double *maxJitter*) [inline]

Get maximum jitter.

Parameters

maxJitter	Maximum jitter in microseconds.

6.1.4.43 void AbstractLayerDescription::setMaxLossRate (const double *maxLossRate*) [inline]

Set maximum loss rate.

Parameters

maxLoss-Rate	Maximum loss rate (out of [0,1]).

6.1.4.44 void AbstractLayerDescription::setMaxTransferDelay (const double *maxDelay*) [inline]

Set maximum transfer delay.

Parameters

maxDelay	Maximum transfer delay in microseconds.

6.1.4.45 void AbstractLayerDescription::setSource (const InternetAddress & *source*) [inline]

Set source address.

Parameters

source	Source address.

6.1.5 Member Data Documentation

6.1.5.1 card64 AbstractLayerDescription::Bandwidth [protected]

6.1.5.2 cardinal AbstractLayerDescription::BufferDelay [protected]

6.1.5.3 InternetFlow AbstractLayerDescription::Destination [protected]

6.1.5.4 cardinal AbstractLayerDescription::Flags [protected]

6.1.5.5 cardinal AbstractLayerDescription::MaxBufferDelay [protected]

6.1.5.6 double AbstractLayerDescription::MaxJitter [protected]

6.1.5.7 double AbstractLayerDescription::MaxLossRate [protected]

6.1.5.8 double AbstractLayerDescription::MaxTransferDelay [protected]

6.1.5.9 cardinal AbstractLayerDescription::PktHeaderSize [protected]

6.1.5.10 cardinal AbstractLayerDescription::PktMaxSize [protected]

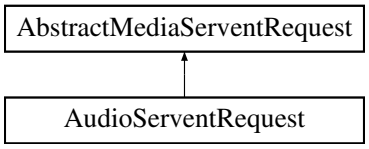
6.1.5.11 InternetAddress AbstractLayerDescription::Source [protected]

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

- abstractlayerdescription.h
- abstractlayerdescription.cc

6.2 AbstractMediaServentRequest Class Reference

Inheritance diagram for AbstractMediaServentRequest:



Public Types

- enum MediaServentMode { MSM_Stop = 0, MSM_Play = 1, MSM_Pause = 2 }

Public Attributes

- card16 SequenceNumber
- card16 PosChgSeqNumber
- MediaServentMode Mode
- integer Speed
- cardinal Encoding
- card64 StartPosition
- card64 RestartPosition
- cardinal BandwidthLimit
- char MediaName [128]

6.2.1 Member Enumeration Documentation

6.2.1.1 enum AbstractMediaServentRequest::MediaServentMode

Enumerator:

MSM_Stop

MSM_Play

MSM_Pause

6.2.2 Member Data Documentation

6.2.2.1 cardinal AbstractMediaServentRequest::BandwidthLimit

6.2.2.2 cardinal AbstractMediaServentRequest::Encoding

6.2.2.3 char AbstractMediaServentRequest::MediaName[128]

6.2.2.4 MediaServentMode AbstractMediaServentRequest::Mode

6.2.2.5 card16 AbstractMediaServentRequest::PosChgSeqNumber

6.2.2.6 card64 AbstractMediaServentRequest::RestartPosition

6.2.2.7 card16 AbstractMediaServentRequest::SequenceNumber

6.2.2.8 integer AbstractMediaServentRequest::Speed

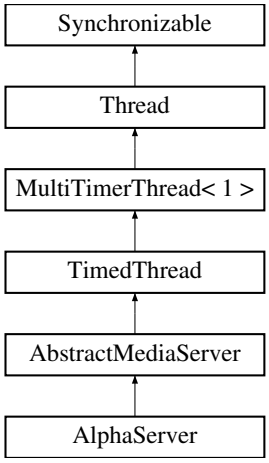
6.2.2.9 card64 AbstractMediaServentRequest::StartPosition

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

- s2.cc

6.3 AbstractMediaServer Class Reference

Inheritance diagram for AbstractMediaServer:



Public Member Functions

- AbstractMediaServer ()
- ~AbstractMediaServer ()

Protected Member Functions

- MediaServent * createServent (const String &identifier, SocketAddress *peer-Address)
- void deleteServent (const String &identifier, const ShutdownReason reason)
- MediaServent * findServent (const String &identifier)
- virtual MediaServent * serventFactory (const String &identifier, SocketAddress *peerAddress)=0

- virtual AbstractMediaServentRequest * createRequest (void *request, const size_t length)=0

Private Member Functions

- void * stop ()
- void timerEvent ()

Private Attributes

- card64 DefaultTimeout
- multimap< const String, MediaServent * > ServentSet

Friends

- class RTPAdaptionLayer

6.3.1 Constructor & Destructor Documentation

6.3.1.1 AbstractMediaServer::AbstractMediaServer ( )

6.3.1.2 AbstractMediaServer::~AbstractMediaServer ( )

6.3.2 Member Function Documentation

6.3.2.1 virtual AbstractMediaServentRequest* AbstractMediaServer::createRequest (void * *request*, const size_t *length*) [protected, pure virtual]

Implemented in AlphaServer.

6.3.2.2 MediaServent * AbstractMediaServer::createServent (const String & *identifier*, SocketAddress * *peerAddress*) [protected]

6.3.2.3 void AbstractMediaServer::deleteServent (const String & *identifier*, const ShutdownReason *reason*) [protected]

6.3.2.4 MediaServent * AbstractMediaServer::findServent (const String & *identifier*) [protected]

6.3.2.5 virtual MediaServent* AbstractMediaServer::serventFactory (const String & *identifier*, SocketAddress * *peerAddress*) [protected, pure virtual]

Implemented in AlphaServer.

6.3.2.6 void * AbstractMediaServer::stop ( ) [private, virtual]

Reimplementation of Thread's stop() method.

See also

Thread::stop

Reimplemented from MultiTimerThread< Timers >.

6.3.2.7 void AbstractMediaServer::timerEvent ( ) [private, virtual]

The virtual timerEvent() method, which contains the timed thread's implementation. It has to be implemented by classes, which inherit TimedThread. This method is called regularly with the given interval.

Implements TimedThread.

6.3.3 Friends And Related Function Documentation

6.3.3.1 friend class RTPAdaptionLayer [friend]

6.3.4 Member Data Documentation

6.3.4.1 card64 AbstractMediaServer::DefaultTimeout [private]

6.3.4.2 multimap<const String,MediaServent*> AbstractMediaServer::ServentSet [private]

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

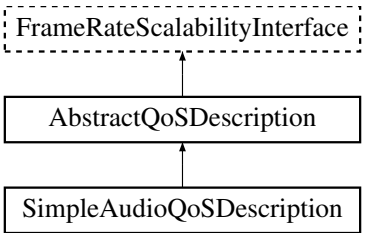
- s2.cc

6.4 AbstractQoSDescription Class Reference

Abstract QoS Description.

`#include <abstractqosdescription.h>`

Inheritance diagram for AbstractQoSDescription:



Public Member Functions

- AbstractQoSDescription ()
- virtual ~AbstractQoSDescription ()
- void initDescription (const double frameRate)
- virtual void updateDescription (const cardinal pktHeaderSize, const cardinal pktMaxSize)=0
- double getFrameRate () const
- double setFrameRate (const double frameRate)
- double getNextFrameRate () const
- double getPrevFrameRate () const
- double getFrameRateScaleFactor () const
- card64 getMinBandwidth () const
- card64 getMaxBandwidth () const
- card64 getPosition () const
- void setPosition (const card64 position)
- virtual cardinal getLayers () const =0
- virtual AbstractLayerDescription * getLayer (const cardinal layer) const =0
- double getResources (ResourceUtilizationPoint &rup) const
- double setResources (const ResourceUtilizationPoint &rup)
- virtual double calculateUtilizationForLayerBandwidths (const double frameRate, const cardinal layers, const card64 *bandwidth) const
- virtual cardinal getPrecomputedResourceUtilizationList (ResourceUtilizationPoint *rup, const card64 bwThreshold, const double utThreshold, const cardinal maxPoints) const =0
- virtual cardinal calculateResourceUtilizationList (ResourceUtilizationPoint *rup, const card64 bwThreshold, const double utThreshold, const cardinal maxPoints) const
- double calculateMaxUtilizationForBandwidth (const card64 totalBandwidth, ResourceUtilizationPoint &rup) const
- virtual void calculateMaxUtilizationForBandwidthArray (const card64 *totalBandwidthArray, ResourceUtilizationPoint *rupArray, const cardinal points) const
- double getWantedUtilization () const
- void setWantedUtilization (const double utilization)
- card64 getMinWantedBandwidth () const
- card64 getMaxWantedBandwidth () const
- void setMinWantedBandwidth (const card64 bandwidth)
- void setMaxWantedBandwidth (const card64 bandwidth)

- int8 getStreamPriority () const
- void setStreamPriority (const int8 priority)
- int8 getSessionPriority () const
- void setSessionPriority (const int8 priority)

Protected Attributes

- double WantedUtilization
- card64 MinWantedBandwidth
- card64 MaxWantedBandwidth
- double FrameRate
- card64 Position
- cardinal PktHeaderSize
- cardinal PktMaxSize
- int8 StreamPriority
- int8 SessionPriority

Private Member Functions

- void doResourceUtilizationIteration (ResourceUtilizationPoint *rup, const card64 bwThreshold, const double utThreshold, double *utilizationCache, card64 *bandwidthCache, const cardinal maxPoints, const cardinal maxCachePoints, const cardinal start, const cardinal end, const card64 startBandwidth, const card64 endBandwidth, const cardinal level, const cardinal maxLevel, cardinal &count) const
- void calculateBandwidthInfo (const cardinal layer, BandwidthInfo &bandwidthInfo) const

6.4.1 Detailed Description

Abstract QoS Description.

This class contains a stream's QoS requirements.

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

6.4.2 Constructor & Destructor Documentation

6.4.2.1 AbstractQoSDescription::AbstractQoSDescription ( )

Constructor.

6.4.2.2 AbstractQoSDescription::~AbstractQoSDescription ( ) [virtual]

Destructor.

6.4.3 Member Function Documentation

6.4.3.1 void AbstractQoSDescription::calculateBandwidthInfo (const cardinal *layer*, BandwidthInfo & *bandwidthInfo*) const [private]

6.4.3.2 double AbstractQoSDescription::calculateMaxUtilizationForBandwidth (const card64 *totalBandwidth*, ResourceUtilizationPoint & *rup*) const [inline]

Calculate maximum utilization for given bandwidth. This is the single-point version of calculateMaxUtilizationForBandwidthArray().

Parameters

total-Bandwidth	Total bandwidth.
rup	ResourceUtilizationPoint reference to store result.

Returns

Utilization.

See also

calculateMaxUtilizationForBandwidthArray

6.4.3.3 void AbstractQoSDescription::calculateMaxUtilizationForBandwidthArray (const card64 * *totalBandwidthArray*, ResourceUtilizationPoint * *rupArray*, const cardinal *points*) const [virtual]

Calculate maximum utilizations for given bandwidth array.

Parameters

total-Bandwidth-Array	Total bandwidth array.
rupArray	ResourceUtilizationPoint array to store results.
points	Number of points in arrays.

6.4.3.4 cardinal AbstractQoSDescription::calculateResourceUtilizationList (ResourceUtilizationPoint * *rup*, const card64 *bwThreshold*, const double *utThreshold*, const cardinal *maxPoints*) const [virtual]

Calculate resource utilization list. To use a precomputed list, call getPrecomputed-ResourceUtilizationList().

Parameters

rup	ResourceUtilizationPoint array capable of storing maxPoints entries.
bwThreshold	Bandwidth threshold.
utThreshold	Utilization threshold.
maxPoints	Maximum number of ResourceUtilizationPoint to generate.

See also

getPrecomputedResourceUtilizationList

6.4.3.5 double AbstractQoSDescription::calculateUtilizationForLayerBandwidths (const double *frameRate*, const cardinal *layers*, const card64 * *bandwidth*) const [virtual]

Calculate utilization for given frame rate and layers bandwidths.

Parameters

frameRate	Frame rate.
layers	Number of layers.
bandwidth	Bandwidth array with entry for each layer.

Returns

Utilization.

6.4.3.6 void AbstractQoSDescription::doResourceUtilizationIteration (ResourceUtilizationPoint * *rup*, const card64 *bwThreshold*, const double *utThreshold*, double * *utilizationCache*, card64 * *bandwidthCache*, const cardinal *maxPoints*, const cardinal *maxCachePoints*, const cardinal *start*, const cardinal *end*, const card64 *startBandwidth*, const card64 *endBandwidth*, const cardinal *level*, const cardinal *maxLevel*, cardinal & *count*) const [private]

6.4.3.7 double AbstractQoSDescription::getFrameRate ( ) const [inline]

Get frame rate.

Returns

Frame rate.

6.4.3.8 double AbstractQoSDescription::getFrameRateScaleFactor ( ) const [inline]

Get frame rate scale factor: (frameRate - MinFrameRate) / (MaxFrameRate - MinFrameRate).

Returns

Frame rate scale factor (out of [0,1]).

6.4.3.9 virtual AbstractLayerDescription* AbstractQoSDescription::getLayer (const cardinal *layer*) const [pure virtual]

Get layer.

Parameters

layer	Layer number.

Returns

Layer.

Implemented in SimpleAudioQoSDescription.

6.4.3.10 virtual cardinal AbstractQoSDescription::getLayers ( ) const [pure virtual]

Get number of layers.

Returns

Number of layers.

Implemented in SimpleAudioQoSDescription.

6.4.3.11 card64 AbstractQoSDescription::getMaxBandwidth ( ) const

Get maximum required total bandwidth.

Returns

Maximum total bandwidth.

6.4.3.12 card64 AbstractQoSDescription::getMaxWantedBandwidth ( ) const

Get maximum wanted bandwidth.

Returns

Maximum wanted bandwidth.

6.4.3.13 card64 AbstractQoSDescription::getMinBandwidth ( ) const

Get minimum required total bandwidth.

Returns

Minimum total bandwidth.

6.4.3.14 card64 AbstractQoSDescription::getMinWantedBandwidth ( ) const

Get minimum wanted bandwidth.

Returns

Minimum wanted bandwidth.

6.4.3.15 double AbstractQoSDescription::getNextFrameRate ( ) const [inline]

Get next higher frame rate.

Returns

Frame rate.

6.4.3.16 card64 AbstractQoSDescription::getPosition ( ) const [inline]

Get position.

Returns

Position.

6.4.3.17 virtual cardinal AbstractQoSDescription::getPrecomputedResourceUtilizationList (ResourceUtilizationPoint * *rup*, const card64 *bwThreshold*, const double *utThreshold*, const cardinal *maxPoints*) const [pure virtual]

Get precomputed resource utilization list. This method tries to use a precomputed list instead of calculating all points like calculateResourceUtilizationList().

Parameters

rup	ResourceUtilizationPoint array capable of storing maxPoints entries.
bwThreshold	Bandwidth threshold.
utThreshold	Utilization threshold.
maxPoints	Maximum number of ResourceUtilizationPoint to generate.

See also

calculateResourceUtilizationList

6.4.3.18 double AbstractQoSDescription::getPrevFrameRate ( ) const [inline]

Get next lower frame rate.

Returns

Frame rate.

6.4.3.19 double AbstractQoSDescription::getResources (ResourceUtilizationPoint & *rup*) const

Get resources.

Parameters

rup	ResourceUtilizationPoint reference to store resources.

Returns

Utilization.

6.4.3.20 int8 AbstractQoSDescription::getSessionPriority ( ) const [inline]

Get session priority.

Returns

Session priority.

6.4.3.21 int8 AbstractQoSDescription::getStreamPriority ( ) const [inline]

Get stream priority.

Returns

Stream priority.

6.4.3.22 double AbstractQoSDescription::getWantedUtilization ( ) const [inline]

Get wanted utilization.

Returns

Wanted utilization.

6.4.3.23 void AbstractQoSDescription::initDescription (const double *frameRate*) [inline]

Initialize description.

Parameters

frameRate	Frame rate.

6.4.3.24 double AbstractQoSDescription::setFrameRate (const double *frameRate*) [inline]

Set frame rate.

Parameters

frameRate	Frame rate.

Returns

Frame rate set.

6.4.3.25 void AbstractQoSDescription::setMaxWantedBandwidth (const card64 *bandwidth*)

Set maximum wanted bandwidth.

Parameters

wanted	bandwidth Maximum wanted bandwidth.

6.4.3.26 void AbstractQoSDescription::setMinWantedBandwidth (const card64 *bandwidth*)

Set minimum wanted bandwidth.

Parameters

wanted	bandwidth Minimum wanted bandwidth.

6.4.3.27 void AbstractQoSDescription::setPosition (const card64 *position*) [inline]

Set position.

Parameters

position	Position.

6.4.3.28 double AbstractQoSDescription::setResources (const ResourceUtilizationPoint & *rup*)

Set resources.

Parameters

rup	ResourceUtilizationPoint reference containing resources.

Returns

Utilization.

6.4.3.29 void AbstractQoSDescription::setSessionPriority (const int8 *priority*) [inline]

Set session priority.

Parameters

priority	Session priority.

6.4.3.30 void AbstractQoSDescription::setStreamPriority (const int8 *priority*) [inline]

Set stream priority.

Parameters

priority	Stream priority.

6.4.3.31 void AbstractQoSDescription::setWantedUtilization (const double *utilization*) [inline]

Set wanted utilization.

Parameters

utilization	Wanted utilization.

6.4.3.32 virtual void AbstractQoSDescription::updateDescription (const cardinal *pktHeaderSize*, const cardinal *pktMaxSize*) [pure virtual]

Update description.

Parameters

pktHeader-Size	Packet header size.
pktMaxSize	Maximum packet size.

Implemented in SimpleAudioQoSDescription.

6.4.4 Member Data Documentation

6.4.4.1 double AbstractQoSDescription::FrameRate [protected]

6.4.4.2 card64 AbstractQoSDescription::MaxWantedBandwidth [protected]

6.4.4.3 card64 AbstractQoSDescription::MinWantedBandwidth [protected]

6.4.4.4 cardinal AbstractQoSDescription::PktHeaderSize [protected]

6.4.4.5 cardinal AbstractQoSDescription::PktMaxSize [protected]

6.4.4.6 card64 AbstractQoSDescription::Position [protected]

6.4.4.7 int8 AbstractQoSDescription::SessionPriority [protected]

6.4.4.8 int8 AbstractQoSDescription::StreamPriority [protected]

6.4.4.9 double AbstractQoSDescription::WantedUtilization [protected]

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

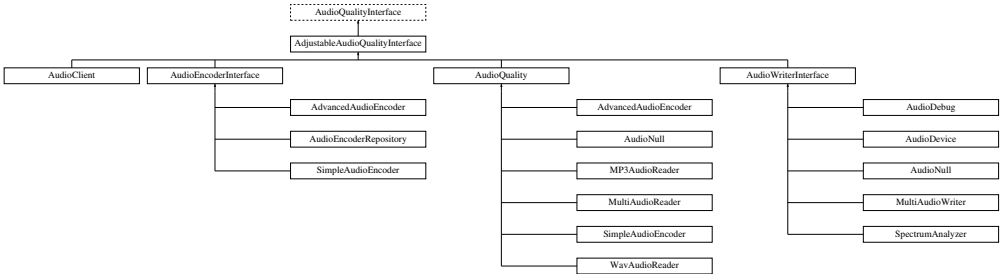
- abstractqosdescription.h
- abstractqosdescription.cc

6.5 AdjustableAudioQualityInterface Class Reference

Adjustable Audio Quality Interface.

`#include <audioqualityinterface.h>`

Inheritance diagram for AdjustableAudioQualityInterface:



Public Member Functions

- virtual card16 setSamplingRate (const card16 samplingRate)=0
- virtual card8 setBits (const card8 bits)=0
- virtual card8 setChannels (const card8 channels)=0
- virtual card16 setByteOrder (const card16 byteOrder)=0
- void setQuality (const AudioQualityInterface &quality)

6.5.1 Detailed Description

Adjustable Audio Quality Interface.

This class is an interface for getting and setting audio quality. It extends AudioQualityInterface with setting functions.

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

6.5.2 Member Function Documentation

6.5.2.1 virtual card8 AdjustableAudioQualityInterface::setBits (const card8 *bits*) [pure virtual]

Set number of bits.

Parameters

samplingRate	New number of bits.

Returns

New number of bits.

Implemented in AudioClient, AudioEncoderRepository, AudioDevice, AudioQuality, MultiAudioWriter, SpectrumAnalyzer, and AudioDebug.

6.5.2.2 virtual card16 AdjustableAudioQualityInterface::setByteOrder (const card16 *byteOrder*) [pure virtual]

Set byte order.

Parameters

byteOrder	New byte order: BIG_ENDIAN, LITTLE_ENDIAN.

Returns

New byte order.

Implemented in AudioClient, AudioEncoderRepository, AudioDevice, AudioQuality, MultiAudioWriter, SpectrumAnalyzer, and AudioDebug.

6.5.2.3 virtual card8 AdjustableAudioQualityInterface::setChannels (const card8 *channels*) [pure virtual]

Set number of channels.

Parameters

samplingRate	New number of channels.

Returns

New number of channels.

Implemented in AudioClient, AudioEncoderRepository, AudioDevice, AudioQuality, MultiAudioWriter, SpectrumAnalyzer, and AudioDebug.

6.5.2.4 void AdjustableAudioQualityInterface::setQuality (const AudioQualityInterface & *quality*) [inline]

Set quality from AudioQualityInterface.

Parameters

quality	AudioQualityInterface.

6.5.2.5 virtual card16 AdjustableAudioQualityInterface::setSamplingRate (const card16 *samplingRate*) [pure virtual]

Set sampling rate.

Parameters

sampling-Rate	New sampling rate.

Returns

New sampling rate.

Implemented in AudioClient, AudioEncoderRepository, AudioDevice, AudioQuality, MultiAudioWriter, SpectrumAnalyzer, and AudioDebug.

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

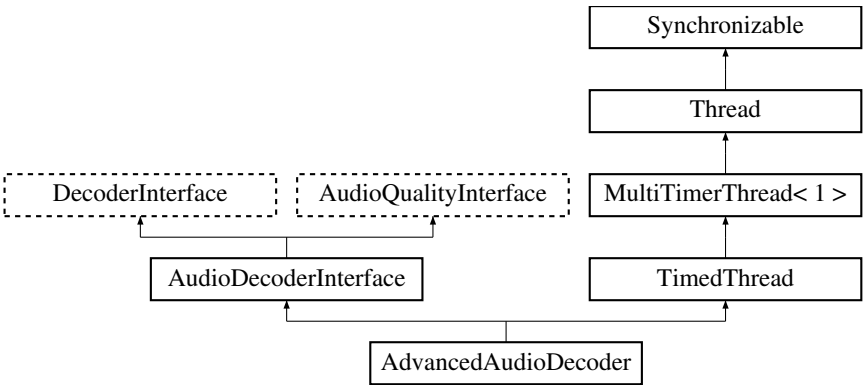
- audioqualityinterface.h

6.6 AdvancedAudioDecoder Class Reference

Advanced Audio Decoder.

`#include <advancedaudiodecoder.h>`

Inheritance diagram for AdvancedAudioDecoder:



Classes

- struct FrameFragment
- struct FrameNode
- struct FrameNodeItem

Public Member Functions

- AdvancedAudioDecoder (AudioWriterInterface *audioWriter)
- ~AdvancedAudioDecoder ()
- const card16 getTypeID () const
- const char * getTypeName () const
- void activate ()
- void deactivate ()
- void reset ()
- void getMediaInfo (MediaInfo &mediaInfo) const
- card8 getErrorCode () const
- card64 getPosition () const
- card64 getMaxPosition () const
- bool checkNextPacket (DecoderPacket *decoderPacket)
- void handleNextPacket (const DecoderPacket *decoderPacket)
- card16 getSamplingRate () const
- card8 getBits () const
- card8 getChannels () const
- card16 getByteOrder () const
- cardinal getBytesPerSecond () const
- cardinal getBitsPerSample () const
- AudioQuality getWantedQuality () const
- void setWantedQuality (const AudioQualityInterface &wantedQuality)

Private Member Functions

- void timerEvent ()
- card64 checkFragmentSeqNum (std::multimap< const card16, FrameFragment * > *set, const card64 last) const
- void deleteFragments (std::multimap< const card16, FrameFragment * > *set)
- FrameFragment * getFragment (std::multimap< const card16, FrameFragment * > *set, const card16 fragmentNumber)

Private Attributes

- std::multiset< FrameNodeItem > FrameSet
- AudioWriterInterface * Device
- AudioQuality WantedQuality
- card64 Position
- card64 MaxPosition
- SeqNumValidator SeqNumber [AdvancedAudioPacket::AdvancedAudioMaxQualityLayers]
- MediaInfo Media
- card16 AudioSamplingRate
- card8 AudioBits
- card8 AudioChannels
- card8 ErrorCode

Static Private Attributes

- static const cardinal FrameBufferSize
- static const card64 BufferCleanUpDifference

6.6.1 Detailed Description

Advanced Audio Decoder.

This class is an advanced audio decoder. It does error correction by using nearly redundant data of left and right channel to "reconstruct" the full data.

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

6.6.2 Constructor & Destructor Documentation

6.6.2.1 AdvancedAudioDecoder::AdvancedAudioDecoder (AudioWriterInterface * *audioWriter*)

Constructor for the audio decoder.

Parameters

audioWriter	AudioReaderInterface for the audio output.

6.6.2.2 AdvancedAudioDecoder::~AdvancedAudioDecoder ( )

Destructor.

6.6.3 Member Function Documentation

6.6.3.1 void AdvancedAudioDecoder::activate ( ) [virtual]

activate() implementation of DecoderInterface.

See also

DecoderInterface::activate

Implements DecoderInterface.

6.6.3.2 card64 AdvancedAudioDecoder::checkFragmentSeqNum (std::multimap< const card16, FrameFragment * > * *set*, const card64 *last*) const [private]

6.6.3.3 bool AdvancedAudioDecoder::checkNextPacket (DecoderPacket * *decoderPacket*) [virtual]

checkNextPacket() implementation of DecoderInterface.

See also

DecoderInterface::checkNextPacket

Implements DecoderInterface.

6.6.3.4 void AdvancedAudioDecoder::deactivate ( ) [virtual]

deactivate() implementation of DecoderInterface.

See also

DecoderInterface::deactivate

Implements DecoderInterface.

6.6.3.5 void AdvancedAudioDecoder::deleteFragments (std::multimap< const card16, FrameFragment * > * *set*) [private]

6.6.3.6 card8 AdvancedAudioDecoder::getBits ( ) const [virtual]

getBits() Implementation of AudioDecoderInterface.

See also

AudioDecoderInterface::getBits

Implements AudioQualityInterface.

6.6.3.7 cardinal AdvancedAudioDecoder::getBitsPerSample ( ) const [virtual]

getBitsPerSample() Implementation of AudioDecoderInterface.

See also

AudioDecoderInterface::getBitsPerSample

Implements AudioQualityInterface.

6.6.3.8 card16 AdvancedAudioDecoder::getByteOrder ( ) const [virtual]

getByteOrder() Implementation of AudioQualityInterface.

See also

AudioQualityInterface::getByteOrder

Implements AudioQualityInterface.

6.6.3.9 cardinal AdvancedAudioDecoder::getBytesPerSecond ( ) const [virtual]

getBytesPerSecond() Implementation of AudioDecoderInterface.

See also

AudioDecoderInterface::getBytesPerSecond

Implements AudioQualityInterface.

6.6.3.10 card8 AdvancedAudioDecoder::getChannels ( ) const [virtual]

getChannels() Implementation of AudioDecoderInterface.

See also

AudioDecoderInterface::getChannels

Implements AudioQualityInterface.

6.6.3.11 card8 AdvancedAudioDecoder::getErrorCode ( ) const [virtual]

getErrorCode() implementation of DecoderInterface.

See also

DecoderInterface::getErrorCode

Implements DecoderInterface.

6.6.3.12 AdvancedAudioDecoder::FrameFragment * AdvancedAudioDecoder::getFragment (std::multimap< const card16, FrameFragment * > * *set*, const card16 *fragmentNumber*) [private]

6.6.3.13 card64 AdvancedAudioDecoder::getMaxPosition ( ) const [virtual]

getMaxPosition() implementation of DecoderInterface.

See also

DecoderInterface::getMaxPosition

Implements DecoderInterface.

6.6.3.14 void AdvancedAudioDecoder::getMediaInfo (MediaInfo & *mediaInfo*) const [virtual]

getMediaInfo() implementation of DecoderInterface.

See also

DecoderInterface::getMediaInfo

Implements DecoderInterface.

6.6.3.15 card64 AdvancedAudioDecoder::getPosition ( ) const [virtual]

getPosition() implementation of DecoderInterface.

See also

DecoderInterface::getPosition

Implements DecoderInterface.

6.6.3.16 card16 AdvancedAudioDecoder::getSamplingRate ( ) const [virtual]

getSamplingRate() Implementation of AudioDecoderInterface.

See also

AudioDecoderInterface::getSamplingRate

Implements AudioQualityInterface.

6.6.3.17 const card16 AdvancedAudioDecoder::getTypeID ( ) const [virtual]

getTypeID() implementation of DecoderInterface.

See also

DecoderInterface::getTypeID

Implements DecoderInterface.

6.6.3.18 const char * AdvancedAudioDecoder::getTypeName ( ) const [virtual]

getTypeName implementation of DecoderInterface.

See also

DecoderInterface::getTypeName

Implements DecoderInterface.

6.6.3.19 AudioQuality AdvancedAudioDecoder::getWantedQuality ( ) const [virtual]

getWantedQuality() implementation of AudioDecoderInterface.

see AudioDecoderInterface::getWantedQuality

Implements AudioDecoderInterface.

6.6.3.20 void AdvancedAudioDecoder::handleNextPacket (const DecoderPacket * *decoderPacket*) [virtual]

handleNextPacket() implementation of DecoderInterface.

See also

DecoderInterface::handleNextPacket

Implements DecoderInterface.

6.6.3.21 void AdvancedAudioDecoder::reset ( ) [virtual]

reset() implementation of DecoderInterface.

See also

DecoderInterface::reset

Implements DecoderInterface.

6.6.3.22 void AdvancedAudioDecoder::setWantedQuality (const AudioQualityInterface & *wantedQuality*) [virtual]

setWantedQuality() implementation of AudioDecoderInterface.

See also

AudioDecoderInterface::setWantedQuality

Implements AudioDecoderInterface.

6.6.3.23 void AdvancedAudioDecoder::timerEvent ( ) [private, virtual]

The virtual timerEvent() method, which contains the timed thread's implementation. It has to be implemented by classes, which inherit TimedThread. This method is called regularly with the given interval.

Implements TimedThread.

6.6.4 Member Data Documentation

6.6.4.1 card8 AdvancedAudioDecoder::AudioBits [private]

6.6.4.2 card8 AdvancedAudioDecoder::AudioChannels [private]

6.6.4.3 card16 AdvancedAudioDecoder::AudioSamplingRate [private]

6.6.4.4 const card64 AdvancedAudioDecoder::BufferCleanUpDifference [static, private]

Initial value:

```
(4 * FrameBufferSize * PositionStepsPerSecond) /
   AdvancedAudioPacket::AdvancedAudioFramesPerSecond
```

6.6.4.5 AudioWriterInterface* AdvancedAudioDecoder::Device [private]

6.6.4.6 card8 AdvancedAudioDecoder::ErrorCode [private]

6.6.4.7 const cardinal AdvancedAudioDecoder::FrameBufferSize [static, private]

Initial value:

```
2 * ((AdvancedAudioPacket::AdvancedAudioFramesPerSecond /
     (16000 / AdvancedAudioPacket::AdvancedAudioMaxTransferDelay)) + 1)
```

6.6.4.8 std::multiset<FrameNodeItem> AdvancedAudioDecoder::FrameSet [private]

6.6.4.9 card64 AdvancedAudioDecoder::MaxPosition [private]

6.6.4.10 MediaInfo AdvancedAudioDecoder::Media [private]

6.6.4.11 card64 AdvancedAudioDecoder::Position [private]

6.6.4.12 SeqNumValidator AdvancedAudioDecoder::SeqNumber[-AdvancedAudioPacket::AdvancedAudioMaxQualityLayers] [private]

6.6.4.13 AudioQuality AdvancedAudioDecoder::WantedQuality [private]

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

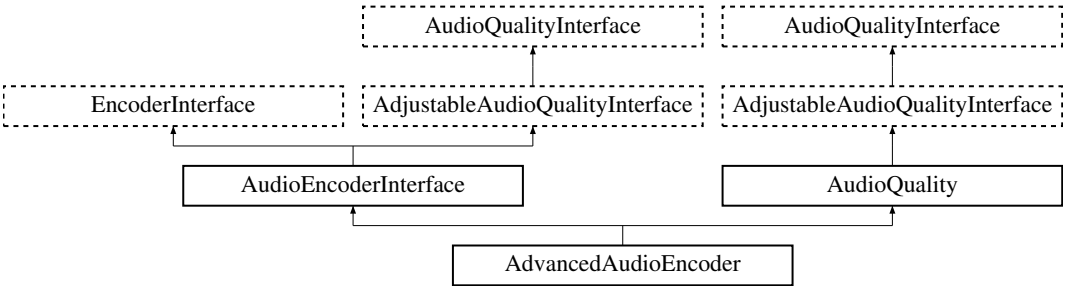
- advancedaudiodecoder.h
- advancedaudiodecoder.cc

6.7 AdvancedAudioEncoder Class Reference

Advanced Audio Encoder.

`#include <advancedaudioencoder.h>`

Inheritance diagram for AdvancedAudioEncoder:



Public Member Functions

- AdvancedAudioEncoder (AudioReaderInterface *audioReader)
- ~AdvancedAudioEncoder ()
- const card16 getTypeID () const
- const char * getTypeName () const
- void activate ()
- void deactivate ()
- void reset ()
- bool checkInterval (card64 &time, bool &newRUList)
- bool prepareNextFrame (const cardinal headerSize, const cardinal maxPacketSize, const cardinal flags)
- cardinal getNextPacket (EncoderPacket *encoderPacket)
- double getFrameRate () const
- AbstractQoSDescription * getQoSDescription (const cardinal pktHeaderSize, const cardinal pktMaxSize, const card64 offset)
- void updateQuality (const AbstractQoSDescription *aqd)

Private Attributes

- AudioReaderInterface * Source
- card64 FramePosition
- card64 FrameMaxPosition
- AudioQuality FrameQualitySetting
- card8 * FrameBufferLL
- card8 * FrameBufferRL
- card8 * FrameBufferLU
- card8 * FrameBufferRU
- cardinal FrameBufferPosLL
- cardinal FrameBufferPosRL
- cardinal FrameBufferPosLU
- cardinal FrameBufferPosRU
- cardinal FrameFragmentLL
- cardinal FrameFragmentRL
- cardinal FrameFragmentLU

- cardinal FrameFragmentRU
- cardinal FrameBufferSizeLL
- cardinal FrameBufferSizeRL
- cardinal FrameBufferSizeLU
- cardinal FrameBufferSizeRU
- cardinal FrameLayerLL
- cardinal FrameLayerRL
- cardinal FrameLayerLU
- cardinal FrameLayerRU
- integer MediaInfoCounter
- card64 TotalByteRateLimit
- card64 ByteRateLimitL1
- card64 ByteRateLimitL2
- card64 ByteRateLimitL3
- cardinal NetworkQualityDecrement
- cardinal SendError
- cardinal SentError
- card8 ErrorCode

6.7.1 Detailed Description

Advanced Audio Encoder.

This class is an advanced audio encoder. It does error correction by using nearly redundant data of left and right channel to "reconstruct" the full data.

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

6.7.2 Constructor & Destructor Documentation

6.7.2.1 AdvancedAudioEncoder::AdvancedAudioEncoder (AudioReaderInterface * *audioReader*)

Constructor for the audio encoder.

Parameters

audioReader	AudioReaderInterface for the audio input.

6.7.2.2 AdvancedAudioEncoder::~AdvancedAudioEncoder ( )

Destructor.

6.7.3 Member Function Documentation

6.7.3.1 void AdvancedAudioEncoder::activate ( ) [virtual]

activate() implementation of EncoderInterface.

See also

EncoderInterface::activate

Implements EncoderInterface.

6.7.3.2 bool AdvancedAudioEncoder::checkInterval (card64 & *time*, bool & *newRUList*) [virtual]

checkInterval() implementation of EncoderInterface.

See also

EncoderInterface::checkInterval

Implements EncoderInterface.

6.7.3.3 void AdvancedAudioEncoder::deactivate ( ) [virtual]

deactivate() implementation of EncoderInterface.

See also

EncoderInterface::deactivate

Implements EncoderInterface.

6.7.3.4 double AdvancedAudioEncoder::getFrameRate ( ) const [virtual]

getFrameRate() implementation of EncoderInterface.

Returns

EncoderInterface::getFrameRate

Implements EncoderInterface.

6.7.3.5 cardinal AdvancedAudioEncoder::getNextPacket (EncoderPacket * *encoderPacket*) [virtual]

getNextPacket() implementation of EncoderInterface.

See also

EncoderInterface::getNextPacket

Implements EncoderInterface.

6.7.3.6 AbstractQoSDescription * AdvancedAudioEncoder::getQoSDescription (const cardinal *pktHeaderSize,* const cardinal *pktMaxSize,* const card64 *offset*) [virtual]

getQoSDescription() implementation of EncoderInterface.

See also

EncoderInterface::getQoSDescription

Implements EncoderInterface.

6.7.3.7 const card16 AdvancedAudioEncoder::getTypeID ( ) const [virtual]

getTypeID() implementation of EncoderInterface.

See also

EncoderInterface::getTypeID

Implements EncoderInterface.

6.7.3.8 const char * AdvancedAudioEncoder::getTypeName ( ) const [virtual]

getTypeName implementation of EncoderInterface.

See also

EncoderInterface::getTypeName

Implements EncoderInterface.

6.7.3.9 bool AdvancedAudioEncoder::prepareNextFrame (const cardinal *headerSize,* const cardinal *maxPacketSize,* const cardinal *flags*) [virtual]

prepareNextFrame() implementation of EncoderInterface.

See also

EncoderInterface::prepareNextFrame

Implements EncoderInterface.

6.7.3.10 void AdvancedAudioEncoder::reset ( ) [virtual]

reset() implementation of EncoderInterface.

See also

EncoderInterface::reset

Implements EncoderInterface.

6.7.3.11 void AdvancedAudioEncoder::updateQuality (const AbstractQoSDescription * *aqd*) [virtual]

updateQuality() implementation of EncoderInterface.

See also

EncoderInterface::updateQuality

Implements EncoderInterface.

6.7.4 Member Data Documentation

6.7.4.1 card64 AdvancedAudioEncoder::ByteRateLimitL1 [private]

6.7.4.2 card64 AdvancedAudioEncoder::ByteRateLimitL2 [private]

6.7.4.3 card64 AdvancedAudioEncoder::ByteRateLimitL3 [private]

6.7.4.4 card8 AdvancedAudioEncoder::ErrorCode [private]

6.7.4.5 card8* AdvancedAudioEncoder::FrameBufferLL [private]

6.7.4.6 card8* AdvancedAudioEncoder::FrameBufferLU [private]

6.7.4.7 cardinal AdvancedAudioEncoder::FrameBufferPosLL [private]

6.7.4.8 cardinal AdvancedAudioEncoder::FrameBufferPosLU [private]

6.7.4.9 cardinal AdvancedAudioEncoder::FrameBufferPosRL [private]

6.7.4.10 **cardinal AdvancedAudioEncoder::FrameBufferPosRU** [private]

6.7.4.11 **card8∗ AdvancedAudioEncoder::FrameBufferRL** [private]

6.7.4.12 **card8∗ AdvancedAudioEncoder::FrameBufferRU** [private]

6.7.4.13 **cardinal AdvancedAudioEncoder::FrameBufferSizeLL** [private]

6.7.4.14 **cardinal AdvancedAudioEncoder::FrameBufferSizeLU** [private]

6.7.4.15 **cardinal AdvancedAudioEncoder::FrameBufferSizeRL** [private]

6.7.4.16 **cardinal AdvancedAudioEncoder::FrameBufferSizeRU** [private]

6.7.4.17 **cardinal AdvancedAudioEncoder::FrameFragmentLL** [private]

6.7.4.18 **cardinal AdvancedAudioEncoder::FrameFragmentLU** [private]

6.7.4.19 **cardinal AdvancedAudioEncoder::FrameFragmentRL** [private]

6.7.4.20 **cardinal AdvancedAudioEncoder::FrameFragmentRU** [private]

6.7.4.21 **cardinal AdvancedAudioEncoder::FrameLayerLL** [private]

6.7.4.22 **cardinal AdvancedAudioEncoder::FrameLayerLU** [private]

6.7.4.23 **cardinal AdvancedAudioEncoder::FrameLayerRL** [private]

6.7.4.24 **cardinal AdvancedAudioEncoder::FrameLayerRU** [private]

6.7.4.25 **card64 AdvancedAudioEncoder::FrameMaxPosition** [private]

6.7.4.26 **card64 AdvancedAudioEncoder::FramePosition** [private]

6.7.4.27 **AudioQuality AdvancedAudioEncoder::FrameQualitySetting** [private]

6.7.4.28 **integer AdvancedAudioEncoder::MediaInfoCounter** [private]

6.7.4.29 **cardinal AdvancedAudioEncoder::NetworkQualityDecrement** [private]

6.7.4.30 **cardinal AdvancedAudioEncoder::SendError** [private]

6.7.4.31 **cardinal AdvancedAudioEncoder::SentError** [private]

6.7.4.32 **AudioReaderInterface∗ AdvancedAudioEncoder::Source** [private]

6.7.4.33 card64 AdvancedAudioEncoder::TotalByteRateLimit [private]

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

- advancedaudioencoder.h
- advancedaudioencoder.cc

6.8 AdvancedAudioPacket Struct Reference

Advanced Audio Packet.

`#include <advancedaudiopacket.h>`

Public Types

- enum AdvancedAudioFlags { AAF_ChannelLeft = (1 << 0), AAF_ChannelRight = (1 << 1), AAF_ByteUpper = (1 << 2), AAF_ByteLower = (1 << 3), AAF_MediaInfo = (1 << 4) }

Public Member Functions

- AdvancedAudioPacket ()
- void translate ()
- void reset ()

Static Public Member Functions

- static AudioQuality calculateQualityForLimits (const AudioQualityInterface &userSetting, const AudioQualityInterface &inputQuality, const card64 totalByteRateLimit, const card64 byteRateLimitL1, const card64 byteRateLimitL2, const card64 byteRateLimitL3, const cardinal networkQualityDecrement, const cardinal headerSize, const cardinal maxPacketSize)
- static cardinal calculateFrameSize (const cardinal inputBytesPerSecond, const cardinal inputFrameSize)
- static cardinal calculateLayers (const AudioQualityInterface &quality)

Public Attributes

- card32 FormatID
- card16 SamplingRate
- card8 Channels
- card8 Bits
- card64 Position
- card64 MaxPosition

- card8 ErrorCode
- card8 Flags
- enum AdvancedAudioPacket::AdvancedAudioFlags __attribute__
- card16 Fragment
- char Data [0]

Static Public Attributes

- static const card16 AdvancedAudioTypeID = 0x2961
- static const char AdvancedAudioTypeName [] = "Advanced Audio Encoding"
- static const card32 AdvancedAudioFormatID = 0x74660000 | AdvancedAudioTypeID
- static const cardinal AdvancedAudioMediaInfoPacketsPerSecond = 2
- static const cardinal AdvancedAudioFramesPerSecond = 35
- static const cardinal AdvancedAudioFrameSize
- static const cardinal AdvancedAudioMaxTransferDelay = 100 * 16
- static const cardinal AdvancedAudioMaxQualityLayers = 3
- static const cardinal AdvancedAudioQualityLevels = AudioQuality::QualityLevels

6.8.1 Detailed Description

Advanced Audio Packet.

This struct defines the packet format for the advanced audio encoder.

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

See also

AdvancedAudioEncoder
AdvancedAudioDecoder

6.8.2 Member Enumeration Documentation

6.8.2.1 enum AdvancedAudioPacket::AdvancedAudioFlags

Emumeration of Flags.

Enumerator:

AAF_ChannelLeft

AAF_ChannelRight

AAF_ByteUpper

AAF_ByteLower

AAF_MediaInfo

6.8.3 Constructor & Destructor Documentation

6.8.3.1 AdvancedAudioPacket::AdvancedAudioPacket ( )

Constructor.

6.8.4 Member Function Documentation

6.8.4.1 cardinal AdvancedAudioPacket::calculateFrameSize (const cardinal *inputBytesPerSecond,* const cardinal *inputFrameSize*) [static]

Calculate output frame size from given input bytes per second and input frame size.

Parameters

inputBytesPerSecond	Input source's bytes per second.
inputFrameSize	Input source's frame size.

Returns

The calculated frame size.

6.8.4.2 cardinal AdvancedAudioPacket::calculateLayers (const AudioQualityInterface & *quality*) [static]

Calculate number of layers for given quality.

Parameters

quality	Quality.

Returns

Number of layers.

6.8.4.3 AudioQuality AdvancedAudioPacket::calculateQualityForLimits (const AudioQualityInterface & *userSetting,* const AudioQualityInterface & *inputQuality,* const card64 *totalByteRateLimit,* const card64 *byteRateLimitL1,* const card64 *byteRateLimitL2,* const card64 *byteRateLimitL3,* const cardinal *networkQualityDecrement,* const cardinal *headerSize,* const cardinal *maxPacketSize*) [static]

Quality calculation for given user quality limited by input quality, byte rate and network quality decrement with given header size (eg. IP + UDP + RTP) and maximum packet size.

Parameters

userSetting	User's quality setting.
inputQuality	Input source's quality.
byteRate-Limit	Byte rate limit.
byteRate-LimitL1	Layer #0 byte rate limit.
byteRate-LimitL2	Layer #1 byte rate limit.
byteRate-LimitL3	Layer #2 byte rate limit.
network-Quality-Decrement	Number of steps for decrement of user's quality.
headerSize	Header size (eg. IP + UDP + RTP). AdvancedAudioPacket size is added automatically.
maxPacket-Size	Maximum packet size.

Returns

The calculated quality.

6.8.4.4 void AdvancedAudioPacket::reset ( )

Reset report.

6.8.4.5 void AdvancedAudioPacket::translate ( )

Translate byte order.

6.8.5 Member Data Documentation

6.8.5.1 enum AdvancedAudioPacket::AdvancedAudioFlags AdvancedAudioPacket::__attribute__

6.8.5.2 const card32 AdvancedAudioPacket::AdvancedAudioFormatID = 0x74660000 | AdvancedAudioTypeID [static]

Advanced Audio Encoding package format ID.

6.8.5.3 const cardinal AdvancedAudioPacket::AdvancedAudioFrameSize [static]

Initial value:

```
44100 * 2 * 2 / AdvancedAudioFramesPerSecond
```

Advanced Audio frame size.

6.8.5.4 const cardinal AdvancedAudioPacket::AdvancedAudioFramesPerSecond = 35 [static]

Advanced Audio frames per second.

6.8.5.5 const cardinal AdvancedAudioPacket::AdvancedAudioMaxQualityLayers = 3 [static]

Advanced Audio maximum quality layers.

6.8.5.6 const cardinal AdvancedAudioPacket::AdvancedAudioMaxTransferDelay = 100 * 16 [static]

Advanced Audio maximum transfer delay.

6.8.5.7 const cardinal AdvancedAudioPacket::AdvancedAudioMediaInfoPackets-PerSecond = 2 [static]

Advanced Audio MediaInfo packets per second.

6.8.5.8 const cardinal AdvancedAudioPacket::AdvancedAudioQualityLevels = AudioQuality::QualityLevels [static]

Advanced Audio quality levels.

6.8.5.9 const card16 AdvancedAudioPacket::AdvancedAudioTypeID = 0x2961 [static]

Type ID for Advanced Audio Encoding.

6.8.5.10 const char AdvancedAudioPacket::AdvancedAudioTypeName = "Advanced Audio Encoding" [static]

Name for Advanced Audio Encoding.

6.8.5.11 card8 AdvancedAudioPacket::Bits

Number of audio bits.

6.8.5.12 card8 AdvancedAudioPacket::Channels

Number of audio channels.

6.8.5.13 char AdvancedAudioPacket::Data[0]

Packet data.

6.8.5.14 card8 AdvancedAudioPacket::ErrorCode

Error code.

6.8.5.15 card8 AdvancedAudioPacket::Flags

Advanced Audio Encoding Flags.

6.8.5.16 card32 AdvancedAudioPacket::FormatID

Packet format ID.

6.8.5.17 card16 AdvancedAudioPacket::Fragment

Fragment number.

6.8.5.18 card64 AdvancedAudioPacket::MaxPosition

Maximum position in nanoseconds.

6.8.5.19 card64 AdvancedAudioPacket::Position

Current position in nanoseconds.

6.8.5.20 card16 AdvancedAudioPacket::SamplingRate

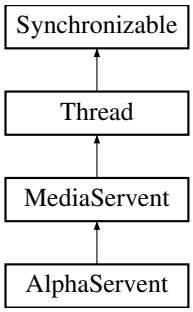
Audio sampling rate.

The documentation for this struct was generated from the following files:

- advancedaudiopacket.h
- advancedaudiopacket.cc

6.9 AlphaServent Class Reference

Inheritance diagram for AlphaServent:



Public Member Functions

- AlphaServent (AbstractMediaServer ∗server, const String &identifier, SocketAddress ∗peerAddress, const integer communicationDomain, const integer socketType, const integer socketProtocol, const SocketAddress ∗∗localAddressArray, const cardinal localAddresses, const cardinal maxPacketSize)
- ∼AlphaServent ()
- bool transmissionErrorOccured ()

Protected Member Functions

- void handleRequest (AbstractMediaServentRequest ∗request)

Private Attributes

- String MediaName

- MultiAudioReader MediaReader
- AudioEncoderRepository EncoderRepository
- RTPSender Sender
- card32 OurSSRC
- cardinal MaxPacketSize
- card64 BandwidthLimit

6.9.1 Constructor & Destructor Documentation

6.9.1.1 AlphaServent::AlphaServent (AbstractMediaServer * *server,* const String & *identifier,* SocketAddress * *peerAddress,* const integer *communicationDomain,* const integer *socketType,* const integer *socketProtocol,* const SocketAddress ** *localAddressArray,* const cardinal *localAddresses,* const cardinal *maxPacketSize*)

6.9.1.2 AlphaServent::~AlphaServent ( )

6.9.2 Member Function Documentation

6.9.2.1 void AlphaServent::handleRequest (AbstractMediaServentRequest * *request*) [protected, virtual]

Implements MediaServent.

6.9.2.2 bool AlphaServent::transmissionErrorOccured ( ) [virtual]

Implements MediaServent.

6.9.3 Member Data Documentation

6.9.3.1 card64 AlphaServent::BandwidthLimit [private]

6.9.3.2 AudioEncoderRepository AlphaServent::EncoderRepository [private]

6.9.3.3 cardinal AlphaServent::MaxPacketSize [private]

6.9.3.4 String AlphaServent::MediaName [private]

6.9.3.5 MultiAudioReader AlphaServent::MediaReader [private]

6.9.3.6 card32 AlphaServent::OurSSRC [private]

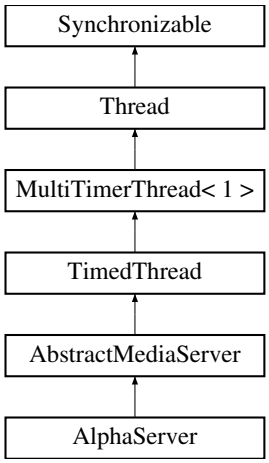
6.9.3.7 RTPSender AlphaServent::Sender [private]

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

- s2.cc

6.10 AlphaServer Class Reference

Inheritance diagram for AlphaServer:



Public Member Functions

- AlphaServer (SocketAddress **localAddressArray=NULL, const cardinal localAddresses=0)
- ~AlphaServer ()
- MediaServent * serventFactory (const String &identifier, SocketAddress *peerAddress)
- AbstractMediaServentRequest * createRequest (void *request, const size_t length)

Private Attributes

- SocketAddress ** LocalAddressArray
- cardinal LocalAddresses
- integer CommunicationDomain
- integer SocketType
- integer SocketProtocol

6.10.1 Constructor & Destructor Documentation

6.10.1.1 AlphaServer::AlphaServer (SocketAddress ** *localAddressArray* = NULL, const cardinal *localAddresses* = 0)

6.10.1.2 AlphaServer::~AlphaServer ( )

6.10.2 Member Function Documentation

6.10.2.1 AbstractMediaServentRequest * AlphaServer::createRequest (void * *request,* const size_t *length*) [virtual]

Implements AbstractMediaServer.

6.10.2.2 MediaServent * AlphaServer::serventFactory (const String & *identifier,* SocketAddress * *peerAddress*) [virtual]

Implements AbstractMediaServer.

6.10.3 Member Data Documentation

6.10.3.1 integer AlphaServer::CommunicationDomain [private]

6.10.3.2 SocketAddress** AlphaServer::LocalAddressArray [private]

6.10.3.3 cardinal AlphaServer::LocalAddresses [private]

6.10.3.4 integer AlphaServer::SocketProtocol [private]

6.10.3.5 integer AlphaServer::SocketType [private]

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

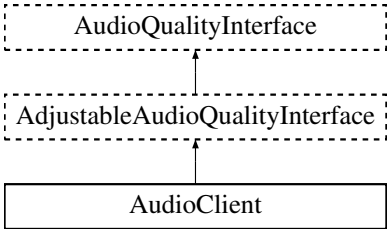
- s2.cc

6.11 AudioClient Class Reference

Audio Client.

`#include <audioclient.h>`

Inheritance diagram for AudioClient:



Public Member Functions

- AudioClient (AudioWriterInterface *audioOutput)
- virtual ~AudioClient ()
- bool play (const char *url)
- bool change (const char *mediaName)
- void stop ()
- card64 getPosition ()
- card64 getMaxPosition () const
- card16 getSamplingRate () const
- card8 getBits () const
- card8 getChannels () const
- card16 getByteOrder () const
- cardinal getBytesPerSecond () const
- cardinal getBitsPerSample () const
- cardinal getRawBytesPerSecond ()
- MediaInfo getMediaInfo () const
- card8 getErrorCode () const
- const char * getEncoding () const
- card32 getBandwidthLimit () const
- card8 getIPVersion () const
- bool playing () const
- String getServerAddressString (const InternetAddress::PrintFormat format=-InternetAddress::PF_Address) const
- String getOurAddressString (const InternetAddress::PrintFormat format=-InternetAddress::PF_Address) const
- cardinal getLayers () const
- card64 getBytesReceived (const cardinal layer=0) const
- card64 getPacketsReceived (const cardinal layer=0) const
- InternetFlow getInternetFlow (const cardinal layer=0) const
- card32 getFlowLabel (const cardinal layer=0) const
- card8 getTrafficClass (const cardinal layer=0) const
- card32 getServerSSRC (const cardinal layer=0) const
- card32 getOurSSRC () const
- card64 getPacketsLost (const cardinal layer=0) const
- double getFractionLost (const cardinal layer=0) const
- double getJitter (const cardinal layer=0) const
- const char * getEncodingName (const cardinal index)
- void setPosition (const card64 position)
- void setPause (const bool on)
- card16 setSamplingRate (const card16 rate)
- card8 setChannels (const card8 channels)
- card8 setBits (const card8 bits)
- card16 setByteOrder (const card16 byteOrder)
- void setEncoding (const cardinal index)
- void setBandwidthLimit (const card32 bandwidthLimit)

Private Member Functions

- void sendCommand (const bool updateRestartPosition=true)

Private Attributes

- AudioWriterInterface * AudioOutput
- RTPReceiver * Receiver
- RTCPSender * Sender
- Socket SenderSocket
- Socket ReceiverSocket
- InternetFlow Flow
- InternetAddress ServerAddress
- InternetAddress OurAddress
- card32 OurSSRC
- std::multimap< const cardinal, AudioDecoderInterface * > DecoderSet
- AudioDecoderRepository Decoders
- AudioClientAppPacket Status
- card64 OldPosition
- card64 ChangeTimeStamp
- bool IsPlaying

Static Private Attributes

- static const card64 RestartPositionUpdateDelay = 5000000

6.11.1 Detailed Description

Audio Client.

This class is an audio client.

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

6.11.2 Constructor & Destructor Documentation

6.11.2.1 AudioClient::AudioClient (AudioWriterInterface * *audioOutput*)

Constructor for a new audio client.

Parameters

audioOutput	AudioWriter to write the output to.

6.11.2.2 AudioClient::~AudioClient ( ) [virtual]

Destructor.

6.11.3 Member Function Documentation

6.11.3.1 bool AudioClient::change (const char * *mediaName*)

Change media of an established connection.

Parameters

mediaName	New media name (e.g. ../AudioFiles/Test2.list)

Returns

true, if play request has been sent to server.

See also

play

6.11.3.2 card32 AudioClient::getBandwidthLimit ( ) const [inline]

Get bandwidth limit.

Returns

Bandwidth limit.

6.11.3.3 card8 AudioClient::getBits ( ) const [virtual]

getBits() Implementation of AudioQualityInterface.

See also

AudioQualityInterface::getBits

Implements AudioQualityInterface.

6.11.3.4 cardinal AudioClient::getBitsPerSample ( ) const [virtual]

getBitsPerSample() Implementation of AudioQualityInterface.

See also

AudioQualityInterface::getBitsPerSample

Implements AudioQualityInterface.

6.11.3.5 card16 AudioClient::getByteOrder ( ) const [virtual]

getByteOrder() Implementation of AudioQualityInterface.

See also

AudioQualityInterface::getByteOrder

Implements AudioQualityInterface.

6.11.3.6 cardinal AudioClient::getBytesPerSecond ( ) const [virtual]

getBytesPerSecond() Implementation of AudioQualityInterface.

See also

AudioQualityInterface::getBytesPerSecond

Implements AudioQualityInterface.

6.11.3.7 card64 AudioClient::getBytesReceived (const cardinal *layer* = 0) const [inline]

Get number of bytes received.

Parameters

layer	Layer number or (cardinal)-1 for sum of all layers.

Returns

Number of bytes received

6.11.3.8 card8 AudioClient::getChannels ( ) const [virtual]

getChannels() Implementation of AudioQualityInterface.

See also

AudioQualityInterface::getChannels

Implements AudioQualityInterface.

6.11.3.9 const char* AudioClient::getEncoding ( ) const [inline]

Get encoding name.

Returns

Encoding name.

6.11.3.10 const char * AudioClient::getEncodingName (const cardinal *index*)

Get encoding name for a given index of the client's decoder repository.

Parameters

index	Repository index.

Returns

Encoding name or NULL, if index is too high.

6.11.3.11 card8 AudioClient::getErrorCode ( ) const [inline]

Get error code.

Returns

Error code.

6.11.3.12 card32 AudioClient::getFlowLabel (const cardinal *layer* = 0) const [inline]

Get flow label of last received packet in given layer.

Parameters

layer	Layer number.

Returns

Flow label.

See also

getLayers

6.11.3.13 double AudioClient::getFractionLost (const cardinal *layer* = 0) const

Get fraction of packets lost for given layer.

Parameters

layer	Layer number.

Returns

Fraction of packets lost.

See also

getLayers

6.11.3.14 InternetFlow AudioClient::getInternetFlow (const cardinal *layer* = 0) const [inline]

Get InternetFlow of last received packet in given layer.

Parameters

layer	Layer number.

Returns

InternetFlow.

See also

getLayers

6.11.3.15 card8 AudioClient::getIPVersion ( ) const

Get IP version.

Returns

IP Version.

6.11.3.16 double AudioClient::getJitter (const cardinal *layer* = 0) const

Get jitter for given layer.

Parameters

layer	Layer number.

Returns

Jitter.

See also

getLayers

6.11.3.17 cardinal AudioClient::getLayers ( ) const [inline]

Get number of layers in last transmission.

6.11.3.18 card64 AudioClient::getMaxPosition ( ) const [inline]

Get maximum media position.

Returns

Maximum position in nanoseconds.

6.11.3.19 MediaInfo AudioClient::getMediaInfo ( ) const

Get MediaInfo.

Returns

MediaInfo.

6.11.3.20 String AudioClient::getOurAddressString (const InternetAddress::PrintFormat *format* = InternetAddress::PF_Address) const

Get client address string.

Parameters

format	Print format.

Returns

Client address.

See also

InternetAddress::PrintFormat

6.11.3.21 card32 AudioClient::getOurSSRC ( ) const [inline]

Get client SSRC.

Returns

Client SSRC.

6.11.3.22 card64 AudioClient::getPacketsLost (const cardinal *layer* = 0) const

Get number of packets lost for given layer.

Parameters

layer	Layer number.

Returns

Number of packets lost.

See also

getLayers

6.11.3.23 card64 AudioClient::getPacketsReceived (const cardinal *layer* = 0) const [inline]

Get number of packets received in given layer.

Parameters

layer	Layer number or (cardinal)-1 for sum of all layers.

Returns

Number of packets received

See also

getLayers

6.11.3.24 card64 AudioClient::getPosition ( )

Get current media position. This will automatically the RestartPosition value in the next AudioClientAppPacket. The server will restart from the current position, if the server is restarted.

Returns

Position in nanoseconds.

6.11.3.25 cardinal AudioClient::getRawBytesPerSecond ( )

Get number of raw bytes (incl. IPv6/UDP/RTP/RTPAudio headers) per second.

Returns

Number of raw bytes per second.

6.11.3.26 card16 AudioClient::getSamplingRate ( ) const [virtual]

getSamplingRate() Implementation of AudioQualityInterface.

See also

AudioQualityInterface::getSamplingRate

Implements AudioQualityInterface.

6.11.3.27 String AudioClient::getServerAddressString (const InternetAddress::PrintFormat *format* = InternetAddress::PF_Address) const

Get server address string.

Parameters

format	Print format.

Returns

Server address.

See also

InternetAddress::PrintFormat

6.11.3.28 card32 AudioClient::getServerSSRC (const cardinal *layer* = 0) const

Get server SSRC for given layer.

Parameters

layer	Layer number.

Returns

Server SSRC.

See also

getLayers

6.11.3.29 card8 AudioClient::getTrafficClass (const cardinal *layer* = 0) const [inline]

Get traffic class of last received packet in given layer.

Parameters

layer	Layer number.

Returns

Traffic class.

See also

getLayers

6.11.3.30 bool AudioClient::play (const char * *url*)

Start playing given media from given server.

Parameters

url	Media URL (e.g. "rtpa+sctp://gaffel:7500/Test1.list").

Returns

true, if play request has been sent to server.

6.11.3.31 bool AudioClient::playing ( ) const [inline]

Check, if audio client is playing.

Returns

true, if client is playing; false otherwise.

6.11.3.32 void AudioClient::sendCommand (const bool *updateRestartPosition* = true) [private]

6.11.3.33 void AudioClient::setBandwidthLimit (const card32 *bandwidthLimit*) [inline]

Set bandwidth limit.

Parameters

bandwidth-Limit	Bandwidth limit.

6.11.3.34 card8 AudioClient::setBits (const card8 *bits*) [virtual]

Set number of audio bits.

Parameters

bits	New number of audio bits.

Implements AdjustableAudioQualityInterface.

6.11.3.35 card16 AudioClient::setByteOrder (const card16 *byteOrder*) [virtual]

Set audio byte order.

Parameters

rate	New audio byte order.

Implements AdjustableAudioQualityInterface.

6.11.3.36 card8 AudioClient::setChannels (const card8 *channels*) [virtual]

Set number of audio channels

Parameters

channels	New number of audio channels.

Implements AdjustableAudioQualityInterface.

6.11.3.37 void AudioClient::setEncoding (const cardinal *index*)

Set encoding by index in client's decoder repository.

Parameters

index	Index in decoder repository.

6.11.3.38 void AudioClient::setPause (const bool *on*)

Set pause.

Parameters

on	true for pause on; false for pause off.

6.11.3.39 void AudioClient::setPosition (const card64 *position*) [inline]

Set media position.

Parameters

position	New media position in nanoseconds.

6.11.3.40 card16 AudioClient::setSamplingRate (const card16 *rate*) [virtual]

Set audio sampling rate.

Parameters

rate	New audio sampling rate.

Implements AdjustableAudioQualityInterface.

6.11.3.41 void AudioClient::stop ( )

Stop playing.

6.11.4 Member Data Documentation

6.11.4.1 AudioWriterInterface* AudioClient::AudioOutput [private]

6.11.4.2 card64 AudioClient::ChangeTimeStamp [private]

6.11.4.3 AudioDecoderRepository AudioClient::Decoders [private]

6.11.4.4 std::multimap<const cardinal,AudioDecoderInterface*> AudioClient::DecoderSet [private]

6.11.4.5 InternetFlow AudioClient::Flow [private]

6.11.4.6 bool AudioClient::IsPlaying [private]

6.11.4.7 card64 AudioClient::OldPosition [private]

6.11.4.8 InternetAddress AudioClient::OurAddress [private]

6.11.4.9 card32 AudioClient::OurSSRC [private]

6.11.4.10 RTPReceiver* AudioClient::Receiver [private]

6.11.4.11 Socket AudioClient::ReceiverSocket [private]

6.11.4.12 const card64 AudioClient::RestartPositionUpdateDelay = 5000000 [static, private]

6.11.4.13 RTCPSender* AudioClient::Sender [private]

6.11.4.14 Socket AudioClient::SenderSocket [private]

6.11.4.15 InternetAddress AudioClient::ServerAddress [private]

6.11.4.16 AudioClientAppPacket AudioClient::Status [private]

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

- audioclient.h
- audioclient.cc

6.12 AudioClientAppPacket Struct Reference

Audio Client RTCP-APP Packet.

`#include <audioclientapppacket.h>`

Public Types

- enum AudioClientAppMode { ACAS_UnknownCommand = 0, ACAS_Play = 1, ACAS_Pause = 2 }

Public Member Functions

- AudioClientAppPacket ()
- void translate ()
- void reset ()

Public Attributes

- enum AudioClientAppPacket::AudioClientAppMode __attribute__
- card32 FormatID
- card16 SequenceNumber
- card16 PosChgSeqNumber
- card16 Status
- card16 SamplingRate
- card8 Channels
- card8 Bits
- card16 Encoding
- card32 BandwidthLimit
- card64 StartPosition
- card64 RestartPosition
- char MediaName [128]

Static Public Attributes

- static const card32 AudioClientFormatID = 0x75003388

6.12.1 Detailed Description

Audio Client RTCP-APP Packet.

This struct defines the packet format for the audio client's RTCP APP-PRIV messages.

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

See also

AudioClient
AudioServer

6.12.2 Member Enumeration Documentation

6.12.2.1 enum AudioClientAppPacket::AudioClientAppMode

Definition of AudioClient commands in APP message.

Enumerator:

ACAS_UnknownCommand

ACAS_Play

ACAS_Pause

6.12.3 Constructor & Destructor Documentation

6.12.3.1 AudioClientAppPacket::AudioClientAppPacket ( )

Constructor.

6.12.4 Member Function Documentation

6.12.4.1 void AudioClientAppPacket::reset ( )

Reset report.

6.12.4.2 void AudioClientAppPacket::translate ( )

Translate byte order.

6.12.5 Member Data Documentation

6.12.5.1 enum AudioClientAppPacket::AudioClientAppMode AudioClientAppPacket::__attribute__

6.12.5.2 const card32 AudioClientAppPacket::AudioClientFormatID = 0x75003388 [static]

Packet ID for AudioClient RTCP APP message.

6.12.5.3 card32 AudioClientAppPacket::BandwidthLimit

Suggested bandwidth or 0xffffffff, if unused.

6.12.5.4 card8 AudioClientAppPacket::Bits

Number of audio bits.

6.12.5.5 card8 AudioClientAppPacket::Channels

Number of audio channels.

6.12.5.6 card16 AudioClientAppPacket::Encoding

Encoding.

6.12.5.7 card32 AudioClientAppPacket::FormatID

Packet ID.

6.12.5.8 char AudioClientAppPacket::MediaName[128]

Media name, e.g. "AudioFiles/Test1.list".

6.12.5.9 card16 AudioClientAppPacket::PosChgSeqNumber

Sequence number for position changes.

6.12.5.10 card64 AudioClientAppPacket::RestartPosition

Position to start from if server has been restarted.

6.12.5.11 card16 AudioClientAppPacket::SamplingRate

Audio sampling rate.

6.12.5.12 card16 AudioClientAppPacket::SequenceNumber

Sequence number.

6.12.5.13 card64 AudioClientAppPacket::StartPosition

Start position in nanoseconds or 0xffff...ff, if unused.

6.12.5.14 card16 AudioClientAppPacket::Status

Client status.

The documentation for this struct was generated from the following files:

- audioclientapppacket.h
- audioclientapppacket.cc

6.13 AudioClientSDESPrivPacket Struct Reference

Audio Client RTCP SDES-PRIV Packet.

`#include <audioclientapppacket.h>`

Public Attributes

- card8 PrefixLength
- char Prefix [7]
- AudioClientAppPacket Status

6.13.1 Detailed Description

Audio Client RTCP SDES-PRIV Packet.

This struct defines the packet format for the audio client's RTCP SDES-PRIV messages.

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

See also

AudioClient
AudioServer

6.13.2 Member Data Documentation

6.13.2.1 char AudioClientSDESPrivPacket::Prefix[7]

6.13.2.2 card8 AudioClientSDESPrivPacket::PrefixLength

6.13.2.3 AudioClientAppPacket AudioClientSDESPrivPacket::Status

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

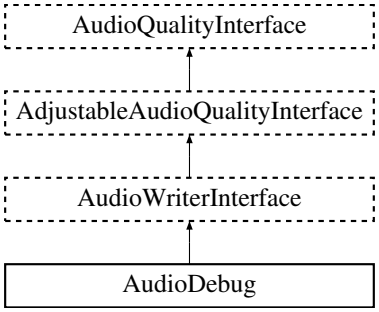
- audioclientapppacket.h

6.14 AudioDebug Class Reference

Audio Debug.

`#include <audiodebug.h>`

Inheritance diagram for AudioDebug:



Public Member Functions

- AudioDebug ()
- ~AudioDebug ()
- card16 getSamplingRate () const
- card8 getBits () const
- card8 getChannels () const

- card16 getByteOrder () const
- cardinal getBytesPerSecond () const
- cardinal getBitsPerSample () const
- card16 setSamplingRate (const card16 samplingRate)
- card8 setBits (const card8 bits)
- card8 setChannels (const card8 channels)
- card16 setByteOrder (const card16 byteOrder)
- bool ready () const
- void sync ()
- bool write (const void *data, const size_t length)

Private Attributes

- card64 LastWriteTimeStamp
- card64 LastPrintTimeStamp
- cardinal BytesWritten
- integer Balance
- card16 AudioSamplingRate
- card8 AudioChannels
- card8 AudioBits
- card16 AudioByteOrder

6.14.1 Detailed Description

Audio Debug.

This class implements AudioWriterInterface for the audio debugger.

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

6.14.2 Constructor & Destructor Documentation

6.14.2.1 AudioDebug::AudioDebug ( )

Constructor.

6.14.2.2 AudioDebug::~AudioDebug ( )

Destructor.

6.14.3 Member Function Documentation

6.14.3.1 card8 AudioDebug::getBits ( ) const [virtual]

getBits() Implementation of AudioQualityInterface.

See also

AudioQualityInterface::getBits

Implements AudioQualityInterface.

6.14.3.2 cardinal AudioDebug::getBitsPerSample ( ) const [virtual]

getBitsPerSample() Implementation of AudioQualityInterface.

See also

AudioQualityInterface::getBitsPerSample

Implements AudioQualityInterface.

6.14.3.3 card16 AudioDebug::getByteOrder ( ) const [virtual]

getByteOrder() Implementation of AudioQualityInterface.

See also

AudioQualityInterface::getByteOrder

Implements AudioQualityInterface.

6.14.3.4 cardinal AudioDebug::getBytesPerSecond ( ) const [virtual]

getBytesPerSecond() Implementation of AudioQualityInterface.

See also

AudioQualityInterface::getBytesPerSecond

Implements AudioQualityInterface.

6.14.3.5 card8 AudioDebug::getChannels ( ) const [virtual]

getChannels() Implementation of AudioQualityInterface.

See also

AudioQualityInterface::getChannels

Implements AudioQualityInterface.

6.14.3.6 card16 AudioDebug::getSamplingRate ( ) const [virtual]

getSamplingRate() Implementation of AudioQualityInterface.

See also

AudioQualityInterface::getSamplingRate

Implements AudioQualityInterface.

6.14.3.7 bool AudioDebug::ready ( ) const [virtual]

ready() implementation of AudioWriterInterface

See also

AudioWriterInterface::ready

Implements AudioWriterInterface.

6.14.3.8 card8 AudioDebug::setBits (const card8 *bits*) [virtual]

setBits() Implementation of AdjustableAudioQualityInterface.

See also

AdjustableAudioQualityInterface::setBits

Implements AdjustableAudioQualityInterface.

6.14.3.9 card16 AudioDebug::setByteOrder (const card16 *byteOrder*) [virtual]

setByteOrder() Implementation of AdjustableAudioQualityInterface.

See also

AdjustableAudioQualityInterface::setByteOrder

Implements AdjustableAudioQualityInterface.

6.14.3.10 card8 AudioDebug::setChannels (const card8 *channels*) [virtual]

setChannels() Implementation of AdjustableAudioQualityInterface.

See also

AdjustableAudioQualityInterface::setChannels

Implements AdjustableAudioQualityInterface.

6.14.3.11 card16 AudioDebug::setSamplingRate (const card16 *samplingRate*) [virtual]

setSamplingRate() Implementation of AdjustableAudioQualityInterface.

See also

AdjustableAudioQualityInterface::setSamplingRate

Implements AdjustableAudioQualityInterface.

6.14.3.12 void AudioDebug::sync ( ) [virtual]

sync() implementation of AudioWriterInterface

See also

AudioWriterInterface::sync

Implements AudioWriterInterface.

6.14.3.13 bool AudioDebug::write (const void * *data*, const size_t *length*) [virtual]

write() implementation of AudioWriterInterface

See also

AudioWriterInterface::write

Implements AudioWriterInterface.

6.14.4 Member Data Documentation

6.14.4.1 card8 AudioDebug::AudioBits [private]

6.14.4.2 card16 AudioDebug::AudioByteOrder [private]

6.14.4.3 card8 AudioDebug::AudioChannels [private]

6.14.4.4 card16 AudioDebug::AudioSamplingRate [private]

6.14.4.5 integer AudioDebug::Balance [private]

6.14.4.6 cardinal AudioDebug::BytesWritten [private]

6.14.4.7 card64 AudioDebug::LastPrintTimeStamp [private]

6.14.4.8 card64 AudioDebug::LastWriteTimeStamp [private]

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

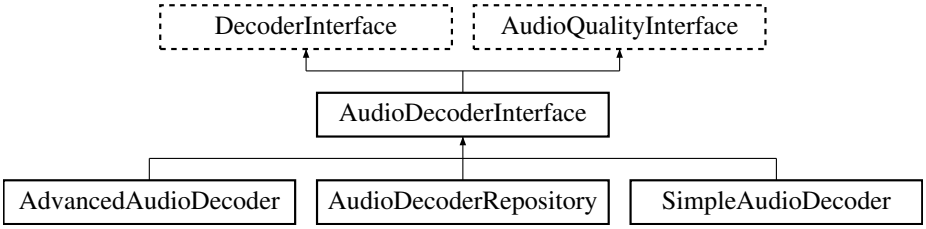
- audiodebug.h
- audiodebug.cc

6.15 AudioDecoderInterface Class Reference

Audio Decoder Interface.

`#include <audiodecoderinterface.h>`

Inheritance diagram for AudioDecoderInterface:



Public Member Functions

- virtual AudioQuality getWantedQuality () const =0
- virtual void setWantedQuality (const AudioQualityInterface &wantedQuality)=0

6.15.1 Detailed Description

Audio Decoder Interface.

This class is the interface for an audio decoder.

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

6.15.2 Member Function Documentation

6.15.2.1 virtual AudioQuality AudioDecoderInterface::getWantedQuality ( ) const [pure virtual]

Get wanted quality. This is the quality wanted in TransportInfo from getTransportInfo().

Returns

Wanted quality.

See also

DecoderInterface::getTransportInfo

Implemented in AudioDecoderRepository, AdvancedAudioDecoder, and SimpleAudioDecoder.

6.15.2.2 virtual void AudioDecoderInterface::setWantedQuality (const AudioQualityInterface & *wantedQuality*) [pure virtual]

Set wanted quality. This is the quality wanted in TransportInfo from getTransportInfo(). Note: This does ∗not∗ tell the sender to modify the quality! This function only sets the wanted quality which is reported by getTransportInfo().

Returns

Wanted quality.

See also

DecoderInterface::getTransportInfo

Implemented in AudioDecoderRepository, AdvancedAudioDecoder, and SimpleAudioDecoder.

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

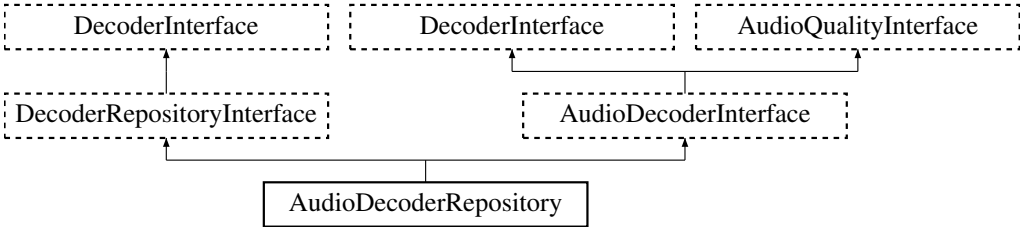
- audiodecoderinterface.h

6.16 AudioDecoderRepository Class Reference

Audio Decoder Repository.

`#include <audiodecoderrepository.h>`

Inheritance diagram for AudioDecoderRepository:



Public Member Functions

- AudioDecoderRepository ()
- ~AudioDecoderRepository ()
- bool addDecoder (AudioDecoderInterface *decoder)
- void removeDecoder (AudioDecoderInterface *decoder)
- bool selectDecoderForTypeID (const card16 typeID)
- void setAutoDelete (const bool on)
- DecoderInterface * getCurrentDecoder () const
- AudioDecoderInterface * getCurrentAudioDecoder () const
- const card16 getTypeID () const
- const char * getTypeName () const
- void activate ()
- void deactivate ()
- void reset ()
- void getMediaInfo (MediaInfo &mediaInfo) const
- card8 getErrorCode () const
- card64 getPosition () const
- card64 getMaxPosition () const
- bool checkNextPacket (DecoderPacket *decoderPacket)
- void handleNextPacket (const DecoderPacket *decoderPacket)
- card8 getChannels () const
- card8 getBits () const
- card16 getSamplingRate () const
- card16 getByteOrder () const
- cardinal getBytesPerSecond () const
- cardinal getBitsPerSample () const
- AudioQuality getWantedQuality () const
- void setWantedQuality (const AudioQualityInterface &wantedQuality)

Private Attributes

- std::multimap< const card16, AudioDecoderInterface * > Repository
- AudioDecoderInterface * Decoder
- bool AutoDelete

6.16.1 Detailed Description

Audio Decoder Repository.

This class is a repository for audio decoders.

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

6.16.2 Constructor & Destructor Documentation

6.16.2.1 AudioDecoderRepository::AudioDecoderRepository ( )

Constructor.

6.16.2.2 AudioDecoderRepository::~AudioDecoderRepository ( )

Destructor.

6.16.3 Member Function Documentation

6.16.3.1 void AudioDecoderRepository::activate ( ) [virtual]

activate() implementation of DecoderInterface.

See also

DecoderInterface::activate

Implements DecoderInterface.

6.16.3.2 bool AudioDecoderRepository::addDecoder (AudioDecoderInterface * *decoder*)

Add audio decoder to repository.

Parameters

decoder	New audio decoder to be added.

Returns

true, if decoder has been added; false, if not.

6.16.3.3 bool AudioDecoderRepository::checkNextPacket (DecoderPacket * *decoderPacket*) [virtual]

checkNextPacket() implementation of DecoderInterface.

See also

DecoderInterface::checkNextPacket

Implements DecoderInterface.

6.16.3.4 void AudioDecoderRepository::deactivate ( ) [virtual]

deactivate() implementation of DecoderInterface.

See also

DecoderInterface::deactivate

Implements DecoderInterface.

6.16.3.5 card8 AudioDecoderRepository::getBits ( ) const [virtual]

getBits() Implementation of AudioDecoderInterface.

See also

AudioDecoderInterface::getBits

Implements AudioQualityInterface.

6.16.3.6 cardinal AudioDecoderRepository::getBitsPerSample ( ) const [virtual]

getBitsPerSample() implementation of AudioDecoderInterface.

See also

AudioDecoderInterface::getBitsPerSample

Implements AudioQualityInterface.

6.16.3.7 card16 AudioDecoderRepository::getByteOrder ( ) const [virtual]

getByteOrder() Implementation of AudioDecoderInterface.

See also

AudioDecoderInterface::getByteOrder

Implements AudioQualityInterface.

6.16.3.8 cardinal AudioDecoderRepository::getBytesPerSecond ( ) const [virtual]

getBytesPerSecond() implementation of AudioDecoderInterface.

See also

AudioDecoderInterface::getBytesPerSecond

Implements AudioQualityInterface.

6.16.3.9 card8 AudioDecoderRepository::getChannels ( ) const [virtual]

getChannels() Implementation of AudioDecoderInterface.

See also

AudioDecoderInterface::getChannels

Implements AudioQualityInterface.

6.16.3.10 AudioDecoderInterface * AudioDecoderRepository::getCurrentAudioDecoder ( ) const

Get AudioDecoderInterface of the current decoder.

Returns

Current decoder's AudioDecoderInterface.

6.16.3.11 DecoderInterface * AudioDecoderRepository::getCurrentDecoder ( ) const [virtual]

getCurrentDecoder() implementation of DecoderRepositoryInterface.

See also

DecoderRepositoryInterface::getCurrentDecoder

Implements DecoderRepositoryInterface.

6.16.3.12 card8 AudioDecoderRepository::getErrorCode ( ) const [virtual]

getErrorCode() implementation of DecoderInterface.

See also

DecoderInterface::getErrorCode

Implements DecoderInterface.

6.16.3.13 card64 AudioDecoderRepository::getMaxPosition ( ) const [virtual]

getMaxPosition() implementation of DecoderInterface.

See also

DecoderInterface::getMaxPosition

Implements DecoderInterface.

6.16.3.14 void AudioDecoderRepository::getMediaInfo (MediaInfo & *mediaInfo*) const [virtual]

getMediaInfo() implementation of DecoderInterface.

See also

DecoderInterface::getMediaInfo

Implements DecoderInterface.

6.16.3.15 card64 AudioDecoderRepository::getPosition ( ) const [virtual]

getPosition() implementation of DecoderInterface.

See also

DecoderInterface::getPosition

Implements DecoderInterface.

6.16.3.16 card16 AudioDecoderRepository::getSamplingRate ( ) const [virtual]

getSamplingRate() Implementation of AudioDecoderInterface.

See also

AudioDecoderInterface::getSamplingRate

Implements AudioQualityInterface.

6.16.3.17 const card16 AudioDecoderRepository::getTypeID ( ) const [virtual]

getTypeID() implementation of DecoderInterface.

See also

DecoderInterface::getTypeID

Implements DecoderInterface.

6.16.3.18 const char * AudioDecoderRepository::getTypeName ( ) const [virtual]

getTypeName implementation of DecoderInterface.

See also

DecoderInterface::getTypeName

Implements DecoderInterface.

6.16.3.19 AudioQuality AudioDecoderRepository::getWantedQuality ( ) const [virtual]

getWantedQuality() implementation of AudioDecoderInterface.

see AudioDecoderInterface::getWantedQuality

Implements AudioDecoderInterface.

6.16.3.20 void AudioDecoderRepository::handleNextPacket (const DecoderPacket * *decoderPacket*) [virtual]

handleNextPacket() implementation of DecoderInterface.

See also

DecoderInterface::handleNextPacket

Implements DecoderInterface.

6.16.3.21 void AudioDecoderRepository::removeDecoder (AudioDecoderInterface * *decoder*)

Remove audio decoder from repository.

Parameters

decoder	Audio decoder to be removed.

6.16.3.22 void AudioDecoderRepository::reset ( ) [virtual]

reset() implementation of DecoderInterface.

See also

DecoderInterface::reset

Implements DecoderInterface.

6.16.3.23 bool AudioDecoderRepository::selectDecoderForTypeID (const card16 *typeID*) [virtual]

selectDecoderForTypeID() implementation of DecoderRepositoryInterface.

See also

DecoderRepositoryInterface::selectDecoderForTypeID

Implements DecoderRepositoryInterface.

6.16.3.24 void AudioDecoderRepository::setAutoDelete (const bool *on*) [inline]

Set AutoDelete mode. If true, all decoders will be deleted with delete operator by the destructor.

6.16.3.25 void AudioDecoderRepository::setWantedQuality (const AudioQualityInterface & *wantedQuality*) [virtual]

setWantedQuality() implementation of AudioDecoderInterface.

See also

AudioDecoderInterface::setWantedQuality

Implements AudioDecoderInterface.

6.16.4 Member Data Documentation

6.16.4.1 bool AudioDecoderRepository::AutoDelete [private]

6.16.4.2 AudioDecoderInterface* AudioDecoderRepository::Decoder [private]

6.16.4.3 std::multimap<const card16,AudioDecoderInterface*> AudioDecoderRepository::Repository [private]

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

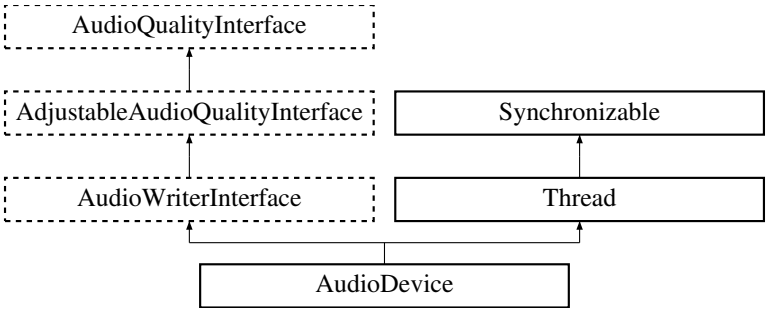
- audiodecoderrepository.h
- audiodecoderrepository.cc

6.17 AudioDevice Class Reference

Audio Device.

`#include <audiodevice.h>`

Inheritance diagram for AudioDevice:



Public Member Functions

- AudioDevice (const char *name="/dev/dsp")
- ~AudioDevice ()
- cardinal getSyncCount () const
- void resetSyncCount ()
- card16 getSamplingRate () const
- card8 getBits () const
- card8 getChannels () const
- card16 getByteOrder () const
- cardinal getBytesPerSecond () const
- cardinal getBitsPerSample () const
- card16 setSamplingRate (const card16 samplingRate)

- card8 setBits (const card8 bits)
- card8 setChannels (const card8 channels)
- card16 setByteOrder (const card16 byteOrder)
- bool ready () const
- void sync ()
- bool write (const void *data, const size_t length)
- cardinal getCurrentCapacity ()

Static Public Attributes

- static const cardinal RingBufferSize = 128 * 1024
- static const cardinal ResizeThresholdPercent = 75
- static const cardinal ResizeModulo = 4

Private Member Functions

- void run ()

Private Attributes

- bool IsReady
- cardinal SyncCount
- cardinal JitterCompensationLatency
- card16 AudioSamplingRate
- card8 AudioBits
- card8 AudioChannels
- card16 AudioByteOrder
- card16 DeviceSamplingRate
- card8 DeviceBits
- card8 DeviceChannels
- card16 DeviceByteOrder
- int DeviceFD
- int DeviceCapabilities
- int DeviceFormats
- integer DeviceBlockSize
- integer DeviceFragmentSize
- integer DeviceOSpace
- RingBuffer Buffer
- cardinal ResizeThreshold
- card64 LastWriteTimeStamp
- integer Balance
- bool IsFillingBuffer

6.17.1 Detailed Description

Audio Device.

This class implements AudioWriterInterface for the audio device.

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

6.17.2 Constructor & Destructor Documentation

6.17.2.1 AudioDevice::AudioDevice (const char * *name* = "/dev/dsp")

Constructor.

Parameters

name	Name of the audio device (normally "/dev/dsp").

6.17.2.2 AudioDevice::~AudioDevice ( )

Destructor.

6.17.3 Member Function Documentation

6.17.3.1 card8 AudioDevice::getBits ( ) const [virtual]

getBits() Implementation of AudioQualityInterface.

See also

AudioQualityInterface::getBits

Implements AudioQualityInterface.

6.17.3.2 cardinal AudioDevice::getBitsPerSample ( ) const [virtual]

getBitsPerSample() Implementation of AudioQualityInterface.

See also

AudioQualityInterface::getBitsPerSample

Implements AudioQualityInterface.

6.17.3.3 card16 AudioDevice::getByteOrder ( ) const [virtual]

getByteOrder() Implementation of AudioQualityInterface.

See also

AudioQualityInterface::getByteOrder

Implements AudioQualityInterface.

6.17.3.4 cardinal AudioDevice::getBytesPerSecond ( ) const [virtual]

getBytesPerSecond() Implementation of AudioQualityInterface.

See also

AudioQualityInterface::getBytesPerSecond

Implements AudioQualityInterface.

6.17.3.5 card8 AudioDevice::getChannels ( ) const [virtual]

getChannels() Implementation of AudioQualityInterface.

See also

AudioQualityInterface::getChannels

Implements AudioQualityInterface.

6.17.3.6 cardinal AudioDevice::getCurrentCapacity ( )

getCurrentCapacity() implementation of AudioWriterInterface

See also

AudioWriterInterface::write

6.17.3.7 card16 AudioDevice::getSamplingRate ( ) const [virtual]

getSamplingRate() Implementation of AudioQualityInterface.

See also

AudioQualityInterface::getSamplingRate

Implements AudioQualityInterface.

6.17.3.8 cardinal AudioDevice::getSyncCount ( ) const [inline]

Get number of times, sync() has been called.

Returns

Number of times, sync() has been called.

See also

AudioWriterInterface::sync

6.17.3.9 bool AudioDevice::ready ( ) const [virtual]

ready() implementation of AudioWriterInterface

See also

AudioWriterInterface::ready

Implements AudioWriterInterface.

6.17.3.10 void AudioDevice::resetSyncCount ( ) [inline]

Reset number of times, sync() has been called.

See also

AudioWriterInterface::sync

6.17.3.11 void AudioDevice::run ( ) [private, virtual]

The virtual run() method, which contains the thread's implementation. It has to be implemented by classes, which inherit Thread.

Implements Thread.

6.17.3.12 card8 AudioDevice::setBits (const card8 *bits*) [virtual]

setBits() Implementation of AdjustableAudioQualityInterface.

See also

AdjustableAudioQualityInterface::setBits

Implements AdjustableAudioQualityInterface.

6.17.3.13 card16 AudioDevice::setByteOrder (const card16 *byteOrder*) [virtual]

setByteOrder() Implementation of AdjustableAudioQualityInterface.

See also

AdjustableAudioQualityInterface::setByteOrder

Implements AdjustableAudioQualityInterface.

6.17.3.14 card8 AudioDevice::setChannels (const card8 *channels*) [virtual]

setChannels() Implementation of AdjustableAudioQualityInterface.

See also

AdjustableAudioQualityInterface::setChannels

Implements AdjustableAudioQualityInterface.

6.17.3.15 card16 AudioDevice::setSamplingRate (const card16 *samplingRate*) [virtual]

setSamplingRate() Implementation of AdjustableAudioQualityInterface.

See also

AdjustableAudioQualityInterface::setSamplingRate

Implements AdjustableAudioQualityInterface.

6.17.3.16 void AudioDevice::sync ( ) [virtual]

sync() implementation of AudioWriterInterface

See also

AudioWriterInterface::sync

Implements AudioWriterInterface.

6.17.3.17 bool AudioDevice::write (const void * *data*, const size_t *length*) [virtual]

write() implementation of AudioWriterInterface

See also

AudioWriterInterface::write

Implements AudioWriterInterface.

6.17.4 Member Data Documentation

6.17.4.1 card8 AudioDevice::AudioBits [private]

6.17.4.2 card16 AudioDevice::AudioByteOrder [private]

6.17.4.3 card8 AudioDevice::AudioChannels [private]

6.17.4.4 card16 AudioDevice::AudioSamplingRate [private]

6.17.4.5 integer AudioDevice::Balance [private]

6.17.4.6 RingBuffer AudioDevice::Buffer [private]

6.17.4.7 card8 AudioDevice::DeviceBits [private]

6.17.4.8 integer AudioDevice::DeviceBlockSize [private]

6.17.4.9 card16 AudioDevice::DeviceByteOrder [private]

6.17.4.10 int AudioDevice::DeviceCapabilities [private]

6.17.4.11 card8 AudioDevice::DeviceChannels [private]

6.17.4.12 int AudioDevice::DeviceFD [private]

6.17.4.13 int AudioDevice::DeviceFormats [private]

6.17.4.14 integer AudioDevice::DeviceFragmentSize [private]

6.17.4.15 integer AudioDevice::DeviceOSpace [private]

6.17.4.16 card16 AudioDevice::DeviceSamplingRate [private]

6.17.4.17 bool AudioDevice::IsFillingBuffer [private]

6.17.4.18 bool AudioDevice::IsReady [private]

6.17.4.19 cardinal AudioDevice::JitterCompensationLatency [private]

6.17.4.20 card64 AudioDevice::LastWriteTimeStamp [private]

6.17.4.21 const cardinal AudioDevice::ResizeModulo = 4 [static]

Buffer resize modulo: The buffer's size will be removed by the fraction of (1/Resize-Modulo) by removing every ResizeModulo-th 32-bit word.

6.17.4.22 cardinal AudioDevice::ResizeThreshold [private]

6.17.4.23 const cardinal AudioDevice::ResizeThresholdPercent = 75 [static]

Buffer fill threshold to Resize (in percent).

6.17.4.24 const cardinal AudioDevice::RingBufferSize = 128 * 1024 [static]

Size of audio ringbuffer in bytes.

6.17.4.25 cardinal AudioDevice::SyncCount [private]

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

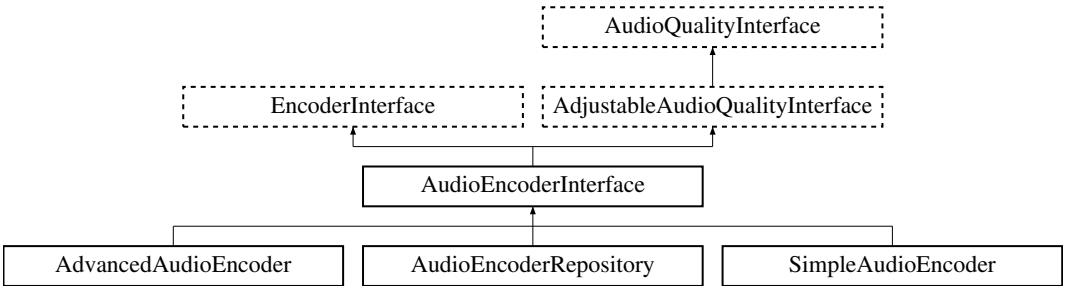
- audiodevice.h
- audiodevice.cc

6.18 AudioEncoderInterface Class Reference

Audio Encoder Interface.

`#include <audioencoderinterface.h>`

Inheritance diagram for AudioEncoderInterface:



6.18.1 Detailed Description

Audio Encoder Interface.

This class is the interface for an audio encoder.

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

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

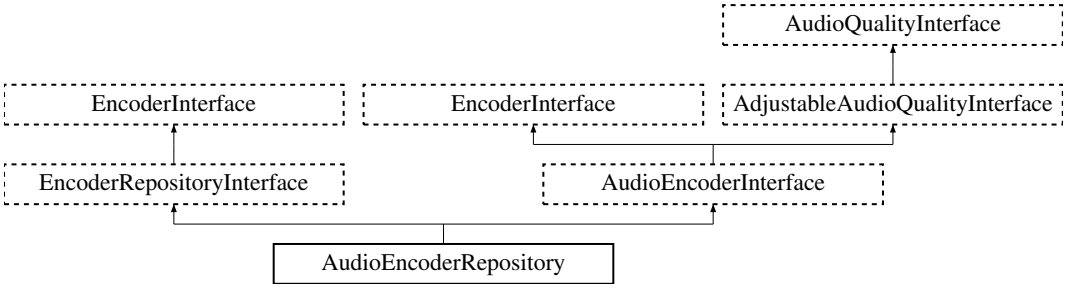
- audioencoderinterface.h

6.19 AudioEncoderRepository Class Reference

Audio Encoder Repository.

`#include <audioencoderrepository.h>`

Inheritance diagram for AudioEncoderRepository:



Public Member Functions

- AudioEncoderRepository ()
- ~AudioEncoderRepository ()
- bool addEncoder (AudioEncoderInterface *encoder)
- void removeEncoder (AudioEncoderInterface *encoder)
- bool selectEncoderForTypeID (const card16 typeID)
- void setAutoDelete (const bool on)
- EncoderInterface * getCurrentEncoder () const
- AudioEncoderInterface * getCurrentAudioEncoder () const
- const card16 getTypeID () const

- const char * getTypeName () const
- void activate ()
- void deactivate ()
- void reset ()
- bool checkInterval (card64 &time, bool &newRUList)
- bool prepareNextFrame (const cardinal headerSize, const cardinal maxPacketSize, const cardinal flags)
- cardinal getNextPacket (EncoderPacket *encoderPacket)
- card16 getSamplingRate () const
- card8 getBits () const
- card8 getChannels () const
- card16 getByteOrder () const
- cardinal getBytesPerSecond () const
- cardinal getBitsPerSample () const
- card16 setSamplingRate (const card16 rate)
- card8 setBits (const card8 bits)
- card8 setChannels (const card8 channels)
- card16 setByteOrder (const card16 byteOrder)
- double getFrameRate () const
- AbstractQoSDescription * getQoSDescription (const cardinal pktHeaderSize, const cardinal pktMaxSize, const card64 offset)
- void updateQuality (const AbstractQoSDescription *aqd)

Private Attributes

- std::multimap< const card16, AudioEncoderInterface * > Repository
- AudioEncoderInterface * Encoder
- bool AutoDelete

6.19.1 Detailed Description

Audio Encoder Repository.

This class is a repository for audio encoders.

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

6.19.2 Constructor & Destructor Documentation

6.19.2.1 AudioEncoderRepository::AudioEncoderRepository ( )

Constructor.

6.19.2.2 AudioEncoderRepository::~AudioEncoderRepository ( )

Destructor.

6.19.3 Member Function Documentation

6.19.3.1 void AudioEncoderRepository::activate ( ) [virtual]

activate() implementation of EncoderInterface.

See also

EncoderInterface::activate

Implements EncoderInterface.

6.19.3.2 bool AudioEncoderRepository::addEncoder (AudioEncoderInterface * *encoder*)

Add audio encoder to repository.

Parameters

encoder	New audio encoder to be added.

Returns

true, if encoder has been added; false, if not.

6.19.3.3 bool AudioEncoderRepository::checkInterval (card64 & *time,* bool & *newRUList*) [virtual]

checkInterval() implementation of EncoderInterface.

See also

EncoderInterface::checkInterval

Implements EncoderInterface.

6.19.3.4 void AudioEncoderRepository::deactivate ( ) [virtual]

deactivate() implementation of EncoderInterface.

See also

EncoderInterface::deactivate

Implements EncoderInterface.

6.19.3.5 card8 AudioEncoderRepository::getBits ( ) const [virtual]

getBits() implementation of AudioEncoderInterface

See also

AudioEncoderInterface::getBits

Implements AudioQualityInterface.

6.19.3.6 cardinal AudioEncoderRepository::getBitsPerSample ( ) const [virtual]

getBitsPerSample() implementation of AudioEncoderInterface.

See also

AudioEncoderInterface::getBitsPerSample

Implements AudioQualityInterface.

6.19.3.7 card16 AudioEncoderRepository::getByteOrder ( ) const [virtual]

getByteOrder() Implementation of AudioEncoderInterface.

See also

AudioEncoderInterface::getByteOrder

Implements AudioQualityInterface.

6.19.3.8 cardinal AudioEncoderRepository::getBytesPerSecond ( ) const [virtual]

getBytesPerSecond() implementation of AudioEncoderInterface.

See also

AudioEncoderInterface::getBytesPerSecond

Implements AudioQualityInterface.

6.19.3.9 card8 AudioEncoderRepository::getChannels ( ) const [virtual]

getChannels() implementation of AudioEncoderInterface

See also

AudioEncoderInterface::getChannels

Implements AudioQualityInterface.

6.19.3.10 AudioEncoderInterface * AudioEncoderRepository::getCurrentAudioEncoder ( ) const

Get AudioEncoderInterface of the current encoder.

Returns

Current encoder's AudioEncoderInterface.

6.19.3.11 EncoderInterface * AudioEncoderRepository::getCurrentEncoder ( ) const [virtual]

getCurrentEncoder() implementation of EncoderRepositoryInterface.

See also

EncoderRepositoryInterface::getCurrentEncoder

Implements EncoderRepositoryInterface.

6.19.3.12 double AudioEncoderRepository::getFrameRate ( ) const [virtual]

getFrameRate() implementation of EncoderInterface.

Returns

EncoderInterface::getFrameRate

Implements EncoderInterface.

6.19.3.13 cardinal AudioEncoderRepository::getNextPacket (EncoderPacket * *encoderPacket*) [virtual]

getNextPacket() implementation of EncoderInterface.

See also

EncoderInterface::getNextPacket

Implements EncoderInterface.

6.19.3.14 AbstractQoSDescription * AudioEncoderRepository::getQoSDescription (const cardinal *pktHeaderSize,* const cardinal *pktMaxSize,* const card64 *offset*) [virtual]

getQoSDescription() implementation of EncoderInterface.

See also

EncoderInterface::getQoSDescription

Implements EncoderInterface.

6.19.3.15 card16 AudioEncoderRepository::getSamplingRate ( ) const [virtual]

getSamplingRate() implementation of AudioEncoderInterface

See also

AudioEncoderInterface::getSamplingRate

Implements AudioQualityInterface.

6.19.3.16 const card16 AudioEncoderRepository::getTypeID ( ) const [virtual]

getTypeID() implementation of EncoderInterface.

See also

EncoderInterface::getTypeID

Implements EncoderInterface.

6.19.3.17 const char * AudioEncoderRepository::getTypeName ( ) const [virtual]

getTypeName implementation of EncoderInterface.

See also

EncoderInterface::getTypeName

Implements EncoderInterface.

6.19.3.18 bool AudioEncoderRepository::prepareNextFrame (const cardinal *headerSize*, const cardinal *maxPacketSize*, const cardinal *flags*) [virtual]

prepareNextFrame() implementation of EncoderInterface.

See also

EncoderInterface::prepareNextFrame

Implements EncoderInterface.

6.19.3.19 void AudioEncoderRepository::removeEncoder (AudioEncoderInterface * *encoder*)

Remove audio encoder from repository.

Parameters

encoder	Audio encoder to be removed.

6.19.3.20 void AudioEncoderRepository::reset ( ) [virtual]

reset() implementation of EncoderInterface.

See also

EncoderInterface::reset

Implements EncoderInterface.

6.19.3.21 bool AudioEncoderRepository::selectEncoderForTypeID (const card16 *typeID*) [virtual]

selectEncoderForTypeID() implementation of EncoderRepositoryInterface.

See also

EncoderRepositoryInterface::selectEncoderForTypeID

Implements EncoderRepositoryInterface.

6.19.3.22 void AudioEncoderRepository::setAutoDelete (const bool *on*) [inline]

Set AutoDelete mode. If true, all encoders will be deleted with delete operator by the destructor.

6.19.3.23 card8 AudioEncoderRepository::setBits (const card8 *bits*) [virtual]

setBits() implementation of AudioEncoderInterface

See also

AudioEncoderInterface::setBits

Implements AdjustableAudioQualityInterface.

6.19.3.24 card16 AudioEncoderRepository::setByteOrder (const card16 *byteOrder*) [virtual]

setByteOrder() Implementation of AudioEncoderInterface.

See also

AudioEncoderInterface::setByteOrder

Implements AdjustableAudioQualityInterface.

6.19.3.25 card8 AudioEncoderRepository::setChannels (const card8 *channels*) [virtual]

setChannels() implementation of AudioEncoderInterface

See also

AudioEncoderInterface::setChannels

Implements AdjustableAudioQualityInterface.

6.19.3.26 card16 AudioEncoderRepository::setSamplingRate (const card16 *rate*) [virtual]

setSamplingRate() implementation of AudioEncoderInterface

See also

AudioEncoderInterface::setSamplingRate

Implements AdjustableAudioQualityInterface.

6.19.3.27 void AudioEncoderRepository::updateQuality (const AbstractQoSDescription * *aqd*) [virtual]

updateQuality() implementation of EncoderInterface.

See also

EncoderInterface::updateQuality

Implements EncoderInterface.

6.19.4 Member Data Documentation

6.19.4.1 bool AudioEncoderRepository::AutoDelete [private]

6.19.4.2 AudioEncoderInterface* AudioEncoderRepository::Encoder [private]

6.19.4.3 std::multimap<const card16,AudioEncoderInterface*> AudioEncoderRepository::Repository [private]

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

- audioencoderrepository.h
- audioencoderrepository.cc

6.20 AudioMixer Class Reference

Audio Mixer.

`#include <audiomixer.h>`

Public Member Functions

- AudioMixer (AudioDevice *audioDevice, int mixerChannel=SOUND_MIXER_PCM, const char *name="/dev/mixer")
- ~AudioMixer ()
- bool ready () const
- bool getVolume (card8 &left, card8 &right)
- bool setVolume (const card8 left, const card8 right)

Private Attributes

- int Device
- int Channel

6.20.1 Detailed Description

Audio Mixer.

This class is an interface to an audio mixer.

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

6.20.2 Constructor & Destructor Documentation

6.20.2.1 AudioMixer::AudioMixer (AudioDevice * *audioDevice,* int *mixerChannel =* SOUND_MIXER_PCM*,* const char * *name =* "/dev/mixer")

Constructor.

Parameters

audioDevice	AudioDevice object.
mixer-Channel	Mixer channel (e.g. SOUND_MIXER_PCM).
name	Mixer device name (e.g. "/dev/mixer").

6.20.2.2 AudioMixer::~AudioMixer ( )

Destructor.

6.20.3 Member Function Documentation

6.20.3.1 bool AudioMixer::getVolume (card8 & *left,* card8 & *right*)

Get volume.

Parameters

left	Volume of left channel.
right	Volume of right channel.

Returns

true, if volume has been written into variables; false otherwise.

6.20.3.2 bool AudioMixer::ready ( ) const [inline]

Check, if mixer is ready.

Returns

true, if mixer is ready; false otherwise.

6.20.3.3 bool AudioMixer::setVolume (const card8 *left*, const card8 *right*)

Set volume.

Parameters

left	Volume of left channel.
right	Volume of right channel.

Returns

true, if volume has been set; false otherwise.

6.20.4 Member Data Documentation

6.20.4.1 int AudioMixer::Channel [private]

6.20.4.2 int AudioMixer::Device [private]

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

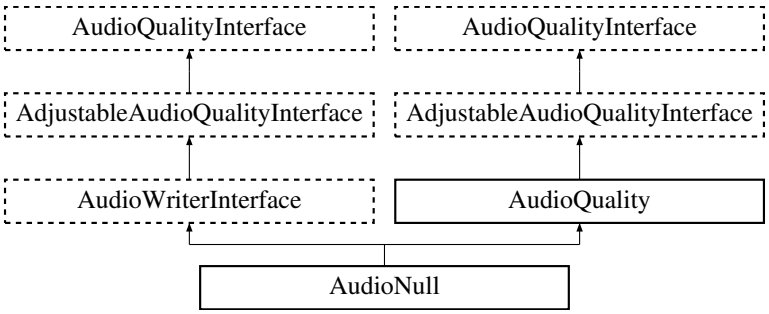
- audiomixer.h
- audiomixer.cc

6.21 AudioNull Class Reference

Audio Null.

`#include <audionull.h>`

Inheritance diagram for AudioNull:



Public Member Functions

- AudioNull ()
- ~AudioNull ()

- bool ready () const
- void sync ()
- bool write (const void *data, const size_t length)

6.21.1 Detailed Description

Audio Null.

This class implements a dummy AudioWriterInterface.

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

6.21.2 Constructor & Destructor Documentation

6.21.2.1 AudioNull::AudioNull ( )

Constructor.

6.21.2.2 AudioNull::~AudioNull ( )

Destructor.

6.21.3 Member Function Documentation

6.21.3.1 bool AudioNull::ready ( ) const [virtual]

ready() implementation of AudioWriterInterface

See also

AudioWriterInterface::ready

Implements AudioWriterInterface.

6.21.3.2 void AudioNull::sync ( ) [virtual]

sync() implementation of AudioWriterInterface

See also

AudioWriterInterface::sync

Implements AudioWriterInterface.

6.21.3.3 bool AudioNull::write (const void * *data*, const size_t *length*) [virtual]

write() implementation of AudioWriterInterface

See also

AudioWriterInterface::write

Implements AudioWriterInterface.

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

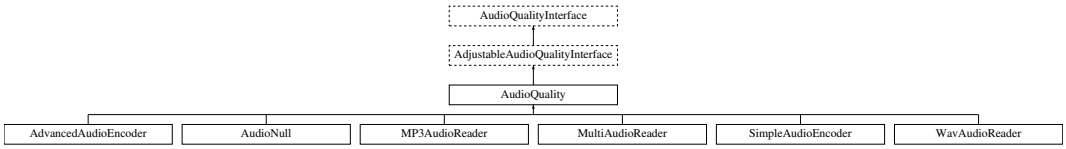
- audionull.h
- audionull.cc

6.22 AudioQuality Class Reference

Audio Quality.

`#include <audioquality.h>`

Inheritance diagram for AudioQuality:



Public Member Functions

- AudioQuality ()
- AudioQuality (const card16 samplingRate, const card8 bits, const card8 channels, const card16 byteOrder=BYTE_ORDER)
- AudioQuality (const AudioQualityInterface &quality)
- card16 getSamplingRate () const
- card8 getBits () const
- card8 getChannels () const
- card16 getByteOrder () const
- cardinal getBytesPerSecond () const
- cardinal getBitsPerSample () const
- card16 setSamplingRate (const card16 samplingRate)
- card8 setBits (const card8 bits)
- card8 setChannels (const card8 channels)
- card16 setByteOrder (const card16 byteOrder)
- bool isLowest () const
- bool isHighest () const

- void increase (const cardinal steps)
- void decrease (const cardinal setps)
- bool prevSamplingRate ()
- bool nextSamplingRate ()
- card64 bytesToTime (const size_t bytes) const
- size_t timeToBytes (const card64 microseconds) const
- AudioQuality & operator= (const AudioQualityInterface &quality)
- AudioQuality operator++ (int)
- AudioQuality operator-- (int)

Static Public Member Functions

- static AudioQuality getQualityForByteRate (const cardinal bps)
- static AudioQuality getRandomQuality (Randomizer *randomizer)

Static Public Attributes

- static const card16 * ValidRatesTable = (const card16*)&_ValidRatesTable
- static const cardinal ValidRates
- static const card8 * ValidBitsTable = (const card8*)&_ValidBitsTable
- static const cardinal ValidBits
- static const card8 * ValidChannelsTable = (const card8*)&_ValidChannelsTable
- static const cardinal ValidChannels
- static const AudioQuality LowestQuality = AudioQuality(4410,4,1,BYTE_ORDER)
- static const AudioQuality HighestQuality = AudioQuality(44100,16,2,BYTE_ORDER)
- static const card16 LowestSamplingRate = 4410
- static const card16 HighestSamplingRate = 44100
- static const card8 LowestBits = 4
- static const card8 HighestBits = 16
- static const card8 LowestChannels = 1
- static const card8 HighestChannels = 2
- static const cardinal QualityLevels = 23

Private Attributes

- card16 SamplingRate
- card8 Bits
- card8 Channels
- card16 ByteOrder

6.22.1 Detailed Description

Audio Quality.

This class manages audio quality.

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

6.22.2 Constructor & Destructor Documentation

6.22.2.1 AudioQuality::AudioQuality ( )

Default constructor.

6.22.2.2 AudioQuality::AudioQuality (const card16 *samplingRate*, const card8 *bits*, const card8 *channels*, const card16 *byteOrder* = BYTE_ORDER)

Constructor for new AudioQuality object with given quality

Parameters

samplingRate	SamplingRate.
bits	Number of bits.
channels	Number of channels.
byteOrder	Byte order: BIG_ENDIAN, LITTLE_ENDIAN.

6.22.2.3 AudioQuality::AudioQuality (const AudioQualityInterface & *quality*)

Constructor for new AudioQuality object from given AudioQualityInterface

Parameters

quality	AudioQualityInterface.

6.22.3 Member Function Documentation

6.22.3.1 card64 AudioQuality::bytesToTime (const size_t *bytes*) const [inline]

Convert bytes to microseconds.

6.22.3.2 void AudioQuality::decrease (const cardinal *setps*)

Decrease quality by given number of steps. The number of steps available is given by QualityLevels constant.

Parameters

steps	Number of steps.

6.22.3.3 card8 AudioQuality::getBits ( ) const [virtual]

getBits() Implementation of AudioQualityInterface.

See also

AudioQualityInterface::getBits

Implements AudioQualityInterface.

6.22.3.4 cardinal AudioQuality::getBitsPerSample ( ) const [virtual]

getBitsPerSample() Implementation of AudioQualityInterface.

See also

AudioQualityInterface::getBitsPerSample

Implements AudioQualityInterface.

6.22.3.5 card16 AudioQuality::getByteOrder ( ) const [virtual]

getByteOrder() Implementation of AudioQualityInterface.

See also

AudioQualityInterface::getByteOrder

Implements AudioQualityInterface.

6.22.3.6 cardinal AudioQuality::getBytesPerSecond ( ) const [virtual]

getBytesPerSecond() Implementation of AudioQualityInterface.

See also

AudioQualityInterface::getBytesPerSecond

Implements AudioQualityInterface.

6.22.3.7 card8 AudioQuality::getChannels ( ) const [virtual]

getChannels() Implementation of AudioQualityInterface.

See also

AudioQualityInterface::getChannels

Implements AudioQualityInterface.

6.22.3.8 AudioQuality AudioQuality::getQualityForByteRate (const cardinal *bps*) [static]

Get maximum audio quality for a given byte rate.

Parameters

bps	Bytes per second.

Returns

AudioQuality.

6.22.3.9 AudioQuality AudioQuality::getRandomQuality (Randomizer * *randomizer*) [static]

Get a random quality setting. All settings have the same probability.

Returns

Random quality setting.

6.22.3.10 card16 AudioQuality::getSamplingRate ( ) const [virtual]

getSamplingRate() Implementation of AudioQualityInterface.

See also

AudioQualityInterface::getSamplingRate

Implements AudioQualityInterface.

6.22.3.11 void AudioQuality::increase (const cardinal *steps*)

Increase quality by given number of steps. The number of steps available is given by QualityLevels constant.

Parameters

steps	Number of steps.

6.22.3.12 bool AudioQuality::isHighest ( ) const [inline]

Check, if quality is highest quality.

Returns

true, is quality is highest; false otherwise.

6.22.3.13 bool AudioQuality::isLowest ( ) const [inline]

Check, if quality is lowest quality.

Returns

true, if quality is lowest; false otherwise.

6.22.3.14 bool AudioQuality::nextSamplingRate ( )

Set sampling rate to next higher value.

Returns

true, if sampling rate has been set; false, if it was already highest.

6.22.3.15 AudioQuality AudioQuality::operator++ (int)

Implementation of ++ operator.

6.22.3.16 AudioQuality AudioQuality::operator-- (int)

Implementation of -- operator.

6.22.3.17 AudioQuality & AudioQuality::operator= (const AudioQualityInterface & *quality*)

Implementation of = operator.

6.22.3.18 bool AudioQuality::prevSamplingRate ( )

Set sampling rate to next lower value.

Returns

true, if sampling rate has been set; false, if it was already lowest.

6.22.3.19 card8 AudioQuality::setBits (const card8 *bits*) [virtual]

setBits() Implementation of AdjustableAudioQualityInterface.

See also

AdjustableAudioQualityInterface::setBits

Implements AdjustableAudioQualityInterface.

6.22.3.20 card16 AudioQuality::setByteOrder (const card16 *byteOrder*) [virtual]

setByteOrder() Implementation of AdjustableAudioQualityInterface.

See also

AdjustableAudioQualityInterface::setByteOrder

Implements AdjustableAudioQualityInterface.

6.22.3.21 card8 AudioQuality::setChannels (const card8 *channels*) [virtual]

setChannels() Implementation of AdjustableAudioQualityInterface.

See also

AdjustableAudioQualityInterface::setChannels

Implements AdjustableAudioQualityInterface.

6.22.3.22 card16 AudioQuality::setSamplingRate (const card16 *samplingRate*) [virtual]

setSamplingRate() Implementation of AdjustableAudioQualityInterface.

See also

AdjustableAudioQualityInterface::setSamplingRate

Implements AdjustableAudioQualityInterface.

6.22.3.23 size_t AudioQuality::timeToBytes (const card64 *microseconds*) const [inline]

Convert microseconds to bytes.

6.22.4 Member Data Documentation

6.22.4.1 card8 AudioQuality::Bits [private]

6.22.4.2 card16 AudioQuality::ByteOrder [private]

6.22.4.3 card8 AudioQuality::Channels [private]

6.22.4.4 const card8 AudioQuality::HighestBits = 16 [static]

Constant for highest number of bits.

6.22.4.5 const card8 AudioQuality::HighestChannels = 2 [static]

Constant for highest number of channels.

6.22.4.6 const AudioQuality AudioQuality::HighestQuality = AudioQuality(44100,16,2,BYTE_ORDER) [static]

Constant for highest quality.

6.22.4.7 const card16 AudioQuality::HighestSamplingRate = 44100 [static]

Constant for highest sampling rate.

6.22.4.8 const card8 AudioQuality::LowestBits = 4 [static]

Constant for lowest number of bits.

6.22.4.9 const card8 AudioQuality::LowestChannels = 1 [static]

Constant for lowest number of channels.

6.22.4.10 const AudioQuality AudioQuality::LowestQuality = AudioQuality(4410,4,1,BYTE_ORDER) [static]

Constant for lowest quality.

6.22.4.11 const card16 AudioQuality::LowestSamplingRate = 4410 [static]

Constant for lowest sampling rate.

6.22.4.12 const cardinal AudioQuality::QualityLevels = 23 [static]

Number of quality levels supported by operator++/operator--.

6.22.4.13 card16 AudioQuality::SamplingRate [private]

6.22.4.14 const cardinal AudioQuality::ValidBits [static]

Initial value:

```
sizeof(_ValidBitsTable) / sizeof(card8)
```

Number of valid bits values in ValidRatesTable.

6.22.4.15 const card8 * AudioQuality::ValidBitsTable = (const card8*)&_ValidBitsTable [static]

Table with valid bit values.

6.22.4.16 const cardinal AudioQuality::ValidChannels [static]

Initial value:

```
sizeof(_ValidChannelsTable) / sizeof(card8)
```

Number of valid channels values in ValidRatesTable.

6.22.4.17 const card8 * AudioQuality::ValidChannelsTable = (const card8*)&_ValidChannelsTable [static]

Table with valid channel values.

6.22.4.18 const cardinal AudioQuality::ValidRates [static]

Initial value:

```
sizeof(_ValidRatesTable) / sizeof(card16)
```

Number of valid sampling rates in ValidRatesTable.

6.22.4.19 const card16 * AudioQuality::ValidRatesTable = (const card16*)&_ValidRatesTable [static]

Table with valid sampling rate values.

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

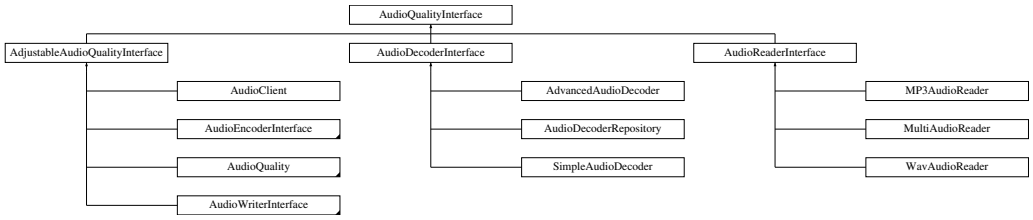
- audioquality.h
- audioquality.cc

6.23 AudioQualityInterface Class Reference

Audio Quality Interface.

`#include <audioqualityinterface.h>`

Inheritance diagram for AudioQualityInterface:



Public Member Functions

- virtual card16 getSamplingRate () const =0
- virtual card8 getBits () const =0
- virtual card8 getChannels () const =0
- virtual card16 getByteOrder () const =0
- virtual cardinal getBytesPerSecond () const =0
- virtual cardinal getBitsPerSample () const =0
- int operator== (const AudioQualityInterface &quality) const
- int operator!= (const AudioQualityInterface &quality) const
- int operator<= (const AudioQualityInterface &quality) const
- int operator< (const AudioQualityInterface &quality) const
- int operator>= (const AudioQualityInterface &quality) const
- int operator> (const AudioQualityInterface &quality) const

6.23.1 Detailed Description

Audio Quality Interface.

This class is an interface for getting audio quality.

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

6.23.2 Member Function Documentation

6.23.2.1 virtual card8 AudioQualityInterface::getBits ( ) const [pure virtual]

Get number of bits.

Returns

Number of bits.

Implemented in AudioDecoderRepository, AudioEncoderRepository, AdvancedAudioDecoder, SimpleAudioDecoder, AudioClient, AudioDevice, MultiAudioWriter, AudioQuality, SpectrumAnalyzer, and AudioDebug.

6.23.2.2 virtual cardinal AudioQualityInterface::getBitsPerSample ( ) const [pure virtual]

Get bits per sample.

Returns

Bits per sample.

Implemented in AudioDecoderRepository, AudioEncoderRepository, AdvancedAudioDecoder, SimpleAudioDecoder, AudioClient, MultiAudioWriter, AudioDevice, AudioQuality, SpectrumAnalyzer, and AudioDebug.

6.23.2.3 virtual card16 AudioQualityInterface::getByteOrder ( ) const [pure virtual]

Get byte order.

Returns

Byte order: BIG_ENDIAN, LITTLE_ENDIAN.

Implemented in AudioDecoderRepository, AudioEncoderRepository, AdvancedAudioDecoder, SimpleAudioDecoder, AudioClient, AudioDevice, MultiAudioWriter, AudioQuality, SpectrumAnalyzer, and AudioDebug.

6.23.2.4 virtual cardinal AudioQualityInterface::getBytesPerSecond ( ) const [pure virtual]

Get bytes per second.

Returns

Bytes per second.

Implemented in AudioDecoderRepository, AudioEncoderRepository, AdvancedAudioDecoder, SimpleAudioDecoder, AudioClient, MultiAudioWriter, AudioDevice, AudioQuality, SpectrumAnalyzer, and AudioDebug.

6.23.2.5 virtual card8 AudioQualityInterface::getChannels ( ) const [pure virtual]

Get number of channels.

Returns

Number of channels.

Implemented in AudioDecoderRepository, AudioEncoderRepository, AdvancedAudioDecoder, SimpleAudioDecoder, AudioClient, AudioDevice, MultiAudioWriter, AudioQuality, SpectrumAnalyzer, and AudioDebug.

6.23.2.6 virtual card16 AudioQualityInterface::getSamplingRate ( ) const [pure virtual]

Get sampling rate.

Returns

Sampling rate.

Implemented in AudioDecoderRepository, AudioEncoderRepository, AdvancedAudioDecoder, SimpleAudioDecoder, AudioClient, AudioDevice, MultiAudioWriter, AudioQuality, SpectrumAnalyzer, and AudioDebug.

6.23.2.7 int AudioQualityInterface::operator!= (const AudioQualityInterface & *quality*) const [inline]

Implementation of != operator.

6.23.2.8 int AudioQualityInterface::operator< (const AudioQualityInterface & *quality*) const [inline]

Implementation of < operator. Note: This operator does not compare byte orders!

6.23.2.9 int AudioQualityInterface::operator<= (const AudioQualityInterface & *quality*) const [inline]

Implementation of <= operator. Note: This operator does not compare byte orders!

6.23.2.10 int AudioQualityInterface::operator== (const AudioQualityInterface & *quality*) const [inline]

Implementation of == operator.

6.23.2.11 int AudioQualityInterface::operator> (const AudioQualityInterface & *quality*) const [inline]

Implementation of > operator. Note: This operator does not compare byte orders!

6.23.2.12 int AudioQualityInterface::operator>= (const AudioQualityInterface & *quality*) const [inline]

Implementation of >= operator. Note: This operator does not compare byte orders!

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

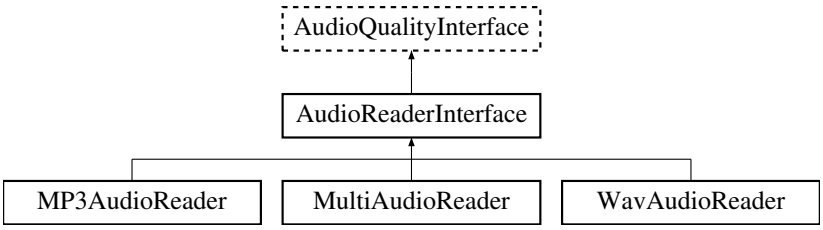
- audioqualityinterface.h

6.24 AudioReaderInterface Class Reference

Audio Reader Interface.

`#include <audioreaderinterface.h>`

Inheritance diagram for AudioReaderInterface:



Public Member Functions

- virtual bool openMedia (const char *name)=0
- virtual void closeMedia ()=0
- virtual bool ready () const =0

- virtual void getMediaInfo (MediaInfo &mediaInfo) const =0
- virtual MediaError getErrorCode () const =0
- virtual card64 getPosition () const =0
- virtual card64 getMaxPosition () const =0
- virtual void setPosition (const card64 position)=0
- virtual cardinal getNextBlock (void *buffer, const cardinal blockSize)=0

6.24.1 Detailed Description

Audio Reader Interface.

This class is the interface for an audio reader.

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

6.24.2 Member Function Documentation

6.24.2.1 virtual void AudioReaderInterface::closeMedia ( ) [pure virtual]

Close media, if opened.

Implemented in MP3AudioReader, MultiAudioReader, and WavAudioReader.

6.24.2.2 virtual MediaError AudioReaderInterface::getErrorCode ( ) const [pure virtual]

Get error code.

Returns

Error code.

Implemented in MP3AudioReader, MultiAudioReader, and WavAudioReader.

6.24.2.3 virtual card64 AudioReaderInterface::getMaxPosition ( ) const [pure virtual]

Get maximum position.

Returns

maximum position in nanoseconds.

Implemented in MP3AudioReader, MultiAudioReader, and WavAudioReader.

6.24.2.4 virtual void AudioReaderInterface::getMediaInfo (MediaInfo & *mediaInfo*) const [pure virtual]

Get MediaInfo.

Parameters

mediaInfo	Reference to store media info.

Implemented in MP3AudioReader, MultiAudioReader, and WavAudioReader.

6.24.2.5 virtual cardinal AudioReaderInterface::getNextBlock (void * *buffer*, const cardinal *blockSize*) [pure virtual]

Read next block. In case of an error, getNextBlock() should return 0 and set ready to false.

Parameters

buffer	Buffer for block to read.
blockSize	Size of block in bytes.

Returns

Number of bytes read.

Implemented in MP3AudioReader, MultiAudioReader, and WavAudioReader.

6.24.2.6 virtual card64 AudioReaderInterface::getPosition ( ) const [pure virtual]

Get current position.

Returns

Position in nanoseconds.

Implemented in MP3AudioReader, MultiAudioReader, and WavAudioReader.

6.24.2.7 virtual bool AudioReaderInterface::openMedia (const char * *name*) [pure virtual]

Open media.

Parameters

name	Name of media, e.g. a file name.

Returns

true, if AudioReader is ready for reading; false otherwise.

Implemented in MP3AudioReader, MultiAudioReader, and WavAudioReader.

6.24.2.8 virtual bool AudioReaderInterface::ready ( ) const [pure virtual]

Check, if AudioReader is ready for reading.

Returns

true, if AudioReader is ready; false otherwise.

Implemented in MP3AudioReader, MultiAudioReader, and WavAudioReader.

6.24.2.9 virtual void AudioReaderInterface::setPosition (const card64 *position*) [pure virtual]

Get position.

Parameters

position	Position in nanoseconds.

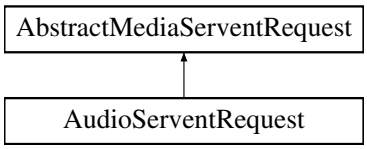
Implemented in MP3AudioReader, MultiAudioReader, and WavAudioReader.

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

- audioreaderinterface.h

6.25 AudioServentRequest Class Reference

Inheritance diagram for AudioServentRequest:



Public Attributes

- card16 SamplingRate
- card8 Channels
- card8 Bits

6.25.1 Member Data Documentation

6.25.1.1 card8 AudioServentRequest::Bits

Number of audio bits.

6.25.1.2 card8 AudioServentRequest::Channels

Number of audio channels.

6.25.1.3 card16 AudioServentRequest::SamplingRate

Audio sampling rate.

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

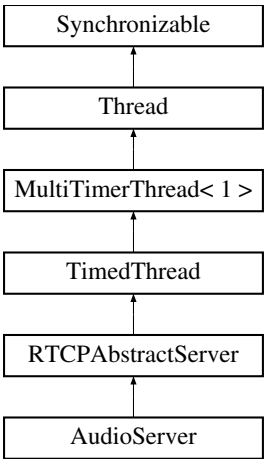
- s2.cc

6.26 AudioServer Class Reference

Audio Server.

`#include <audioserver.h>`

Inheritance diagram for AudioServer:



Classes

- struct User

Public Member Functions

- AudioServer (QoSManagerInterface *qosManager=NULL, const cardinal maxPacketSize=1500, const bool useSCTP=false)
- ~AudioServer ()
- card32 getOurSSRC () const
- bool getLossScalability () const
- void setLossScalability (const bool on)
- cardinal getMaxPacketSize () const
- cardinal setMaxPacketSize (const cardinal size)
- void outOfMemoryWarning ()
- void * newClient (RTCPAbstractServer::Client *client, const char *cname)
- void deleteClient (RTCPAbstractServer::Client *client, const DeleteReason reason)
- bool checkClient (RTCPAbstractServer::Client *client)
- void appMessage (RTCPAbstractServer::Client *client, const char *name, void *data, const cardinal dataLength)
- void sdesMessage (RTCPAbstractServer::Client *client, const card8 type, char *data, const cardinal length)
- void receiverReport (RTCPAbstractServer::Client *client, RTCPReceptionReportBlock *report, const cardinal layer)
- void userCommand (RTCPAbstractServer::Client *client, User *user, AudioClientAppPacket *app)
- void managementUpdate (RTCPAbstractServer::Client *client, User *user)

Private Attributes

- QoSManagerInterface * QoSMgr
- std::multimap< const cardinal, User * > UserSet
- Synchronizable UserSetSync
- cardinal MaxPacketSize
- card32 OurSSRC
- bool LossScalability
- bool UseSCTP

6.26.1 Detailed Description

Audio Server.

This class is an audio server based on RTCPAbstractServer

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

6.26.2 Constructor & Destructor Documentation

6.26.2.1 AudioServer::AudioServer (QoSManagerInterface * *qosManager* = NULL, const cardinal *maxPacketSize* = 1500, const bool *useSCTP* = false)

Constructor for new AudioServer.

Parameters

qosManager	QoS manager.
maxPacket-Size	Maximum packet size.
useSCTP	true to use SCTP instead of UDP; false otherwise.

6.26.2.2 AudioServer::~AudioServer ( )

Destructor.

6.26.3 Member Function Documentation

6.26.3.1 void AudioServer::appMessage (RTCPAbstractServer::Client * *client*, const char * *name*, void * *data*, const cardinal *dataLength*) [virtual]

appMessage() implementation of RTCPAbstractServer.

See also

RTCPAbstractServer::appMessage

Implements RTCPAbstractServer.

6.26.3.2 bool AudioServer::checkClient (RTCPAbstractServer::Client * *client*) [virtual]

checkClient() implementation of RTCPAbstractServer.

See also

RTCPAbstractServer::checkClient

Implements RTCPAbstractServer.

6.26.3.3 void AudioServer::deleteClient (RTCPAbstractServer::Client * *client*, const DeleteReason *reason*) [virtual]

deleteClient() implementation of RTCPAbstractServer.

See also

RTCPAbstractServer::deleteClient

Implements RTCPAbstractServer.

6.26.3.4 bool AudioServer::getLossScalability ( ) const [inline]

Get loss scalibility setting.

Returns

true, if loss scalability is on; false otherwise.

6.26.3.5 cardinal AudioServer::getMaxPacketSize ( ) const [inline]

Get maximum packet size.

Returns

Maximum packet size.

6.26.3.6 card32 AudioServer::getOurSSRC ( ) const

Get client SSRC.

Returns

Client SSRC.

6.26.3.7 void AudioServer::managementUpdate (RTCPAbstractServer::Client * *client,* User * *user*)

Update QoS/congestion management.

Parameters

client	Client to do congestion for.
user	User data.

6.26.3.8 void * AudioServer::newClient (RTCPAbstractServer::Client * *client,* const char * *cname*) [virtual]

newClient() implementation of RTCPAbstractServer.

See also

RTCPAbstractServer::newClient

Implements RTCPAbstractServer.

6.26.3.9 void AudioServer::outOfMemoryWarning ( ) [virtual]

outOfMemoryWarning() implementation of RTCPAbstractServer.

Reimplemented from RTCPAbstractServer.

6.26.3.10 void AudioServer::receiverReport (RTCPAbstractServer::Client * *client*, RTCPReceptionReportBlock * *report*, const cardinal *layer*) [virtual]

receiverReport() implementation of RTCPAbstractServer.

See also

RTCPAbstractServer::receiverReport

Implements RTCPAbstractServer.

6.26.3.11 void AudioServer::sdesMessage (RTCPAbstractServer::Client * *client*, const card8 *type*, char * *data*, const cardinal *length*) [virtual]

sdesMessage() implementation of RTCPAbstractServer.

See also

RTCPAbstractServer::sdesMessage

Implements RTCPAbstractServer.

6.26.3.12 void AudioServer::setLossScalability (const bool *on*) [inline]

Set loss scalibility setting.

Parameters

on	true, if to set loss scalability on; false otherwise.

6.26.3.13 cardinal AudioServer::setMaxPacketSize (const cardinal *size*) [inline]

Set maximum packet size.

Parameters

size	Maximum packet size.

Returns

Maximum packet size set.

6.26.3.14 void AudioServer::userCommand (RTCPAbstractServer::Client * *client*, User * *user*, AudioClientAppPacket * *app*)

Execute commands given in AudioClientAppPacket.

Parameters

client	Client.
user	User.
app	AudioClientApp message.

6.26.4 Member Data Documentation

6.26.4.1 bool AudioServer::LossScalability [private]

6.26.4.2 cardinal AudioServer::MaxPacketSize [private]

6.26.4.3 card32 AudioServer::OurSSRC [private]

6.26.4.4 QoSManagerInterface* AudioServer::QoSMgr [private]

6.26.4.5 std::multimap<const cardinal,User*> AudioServer::UserSet [private]

6.26.4.6 Synchronizable AudioServer::UserSetSync [private]

6.26.4.7 bool AudioServer::UseSCTP [private]

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

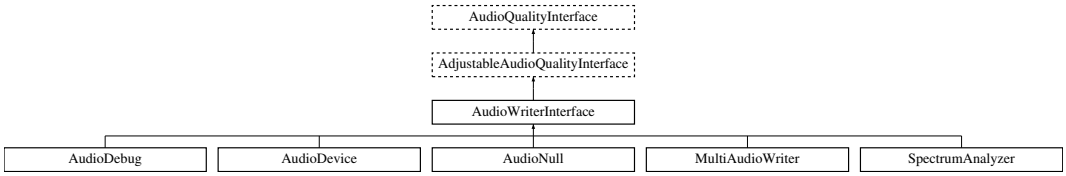
- audioserver.h
- audioserver.cc

6.27 AudioWriterInterface Class Reference

Audio Writer Interface.

`#include <audiowriterinterface.h>`

Inheritance diagram for AudioWriterInterface:



Public Member Functions

- virtual bool ready () const =0
- virtual void sync ()=0
- virtual bool write (const void *data, const size_t length)=0

6.27.1 Detailed Description

Audio Writer Interface.

This class is the interface for an audio writer.

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

6.27.2 Member Function Documentation

6.27.2.1 virtual bool AudioWriterInterface::ready ( ) const [pure virtual]

Check, if AudioWriter is ready for writing.

Returns

true, if AudioWriter is ready; false otherwise.

Implemented in AudioDevice, MultiAudioWriter, SpectrumAnalyzer, AudioDebug, and AudioNull.

6.27.2.2 virtual void AudioWriterInterface::sync ( ) [pure virtual]

Reset the writer. All data in the output buffer should be removed without writing. Usage example: AudioDevice sends ioctl SNDCTL_DSP_SYNC.

Implemented in AudioDevice, MultiAudioWriter, SpectrumAnalyzer, AudioDebug, and AudioNull.

6.27.2.3 virtual bool AudioWriterInterface::write (const void * *data,* const size_t *length*) [pure virtual]

Write data.

Parameters

data	Data to be written.

Returns

length Length of data in bytes.

Implemented in AudioDevice, MultiAudioWriter, SpectrumAnalyzer, AudioDebug, and AudioNull.

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

- audiowriterinterface.h

6.28 BandwidthInfo Struct Reference

Bandwidth Info.

`#include <bandwidthinfo.h>`

Public Member Functions

- void reset ()
- int operator== (const BandwidthInfo &rup) const
- int operator!= (const BandwidthInfo &rup) const

Public Attributes

- cardinal BufferDelay
- card64 BytesPerSecond
- cardinal PacketsPerSecond
- double MaxTransferDelay
- double MaxLossRate
- double MaxJitter

6.28.1 Detailed Description

Bandwidth Info.

This is a description of bandwidth requirements.

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

6.28.2 Member Function Documentation

6.28.2.1 int BandwidthInfo::operator!= (const BandwidthInfo & *rup*) const [inline]

Operator "!=".

6.28.2.2 int BandwidthInfo::operator== (const BandwidthInfo & *rup*) const [inline]

Operator "==".

6.28.2.3 void BandwidthInfo::reset ( )

Reset.

6.28.3 Member Data Documentation

6.28.3.1 cardinal BandwidthInfo::BufferDelay

Buffer delay in microseconds.

6.28.3.2 card64 BandwidthInfo::BytesPerSecond

Bytes per second.

6.28.3.3 double BandwidthInfo::MaxJitter

Maximum jitter.

6.28.3.4 double BandwidthInfo::MaxLossRate

Maximum loss rate.

6.28.3.5 double BandwidthInfo::MaxTransferDelay

Maximum transfer delay.

6.28.3.6 cardinal BandwidthInfo::PacketsPerSecond

Packets per second.

The documentation for this struct was generated from the following files:

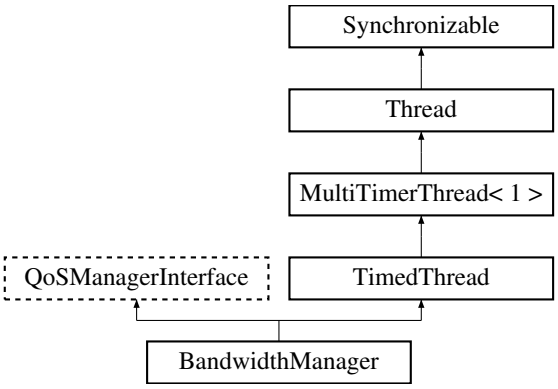
- bandwidthinfo.h
- bandwidthinfo.cc

6.29 BandwidthManager Class Reference

Bandwidth Manager.

`#include <bandwidthmanager.h>`

Inheritance diagram for BandwidthManager:



Public Member Functions

- BandwidthManager (ServiceLevelAgreement ∗sla, RoundTripTimePinger ∗rttp)
- ∼BandwidthManager ()
- void addStream (ManagedStreamInterface ∗stream, const cardinal sessionID=0, const char ∗name=NULL)
- void removeStream (ManagedStreamInterface ∗stream)
- void updateStream (ManagedStreamInterface ∗stream)
- void forceCompleteRemapping ()
- void intervalChangeEvent (ManagedStreamInterface ∗stream, const bool isNew, const card64 when, const bool newRUList)
- void reportEvent (ManagedStreamInterface ∗stream, const RTCPReception-ReportBlock ∗report, const cardinal layer)
- void bufferFlushEvent (ManagedStreamInterface ∗stream, const cardinal layer)
- void timerEvent ()
- void setLogStream (std::ostream ∗logStream)

- void getPartialRemapping (bool &enabled, double &reservedPortion, double &utilizationTolerance, double &maxRemappingInterval) const
- void setPartialRemapping (const bool enabled, const double reservedPortion, const double utilizationTolerance, const double maxRemappingInterval)
- void getFairness (double &fairnessSession, double &fairnessStream) const
- void setFairness (const double fairnessSession, const double fairnessStream)
- void getQoSOptimizationParameters (cardinal &maxRUPoints, double &utilizationThreshold, card64 &bandwidthThreshold, double &systemDelayTolerance, bool &unlayeredAllocation) const
- void setQoSOptimizationParameters (const cardinal maxRUPoints, const double utilizationThreshold, const card64 bandwidthThreshold, const double systemDelayTolerance, const bool unlayeredAllocation)
- void updateReservation (StreamDescription *streamDescription)

Public Attributes

- card64 TotalAvailableBandwidth
- card64 ClassAvailableBandwidthArray [TrafficClassValues::MaxValues]
- card64 TotalBandwidth
- card64 ClassBandwidthArray [TrafficClassValues::MaxValues]
- int64 SLAUpdateRecommendation [TrafficClassValues::MaxValues]
- card64 StreamIDGenerator
- card64 LastCompleteRemapping
- card64 LastCompleteRemappingDuration
- cardinal CompleteRemappings
- cardinal PartialRemappings
- cardinal TotalBufferFlushes
- std::multimap < ManagedStreamInterface *, StreamDescription * > StreamSet
- std::multimap< cardinal, SessionDescription * > SessionSet
- ServiceLevelAgreement * SLA
- cardinal Streams
- cardinal Sessions
- cardinal MaxRUPoints
- double UtilizationThreshold
- card64 BandwidthThreshold
- double SystemDelayTolerance
- double FairnessSession
- double FairnessStream
- double AlphaLossRate
- double AlphaJitter
- double PartialRemappingPortion
- double PartialRemappingUtilizationTolerance
- card64 MaxRemappingInterval
- bool EnablePartialRemappings
- bool UnlayeredAllocation

Static Public Attributes

- static card64 SimulatorTime = 0

Private Member Functions

- cardinal calculateSessionMultiPoints (SessionDescription *session, const cardinal offset, const cardinal lastPoint, ResourceUtilizationMultiPoint *rumpList)
- void getRoundTripTimes (StreamDescription *sd)
- double getPriorityFactor (const int8 streamPriority) const
- double getResourcePart (const ResourceUtilizationSimplePoint &rup) const
- double getResourcePart (const ResourceUtilizationMultiPoint &rump) const
- double getStreamSortingValue (const ResourceUtilizationSimplePoint &rup) const
- double getSessionSortingValue (const ResourceUtilizationMultiPoint &rump) const
- bool tryAllocation (ResourceUtilizationMultiPoint &rump, const card64 bandwidthLimit=(card64)-1)
- void doAllocationTrials (ResourceUtilizationMultiPoint *rumpList, const cardinal points, const card64 bandwidthLimit=(card64)-1)
- bool doPartialRemapping (StreamDescription *streamDescription)
- void doCompleteRemapping ()

Static Private Member Functions

- static void smoothedUpdate (double &value, const double measured, const double alpha)

Private Attributes

- RoundTripTimePinger * RTTP
- std::ostream * Log
- card64 LogStartupTimeStamp
- bool Changed

6.29.1 Detailed Description

Bandwidth Manager.

This is the bandwidth manager.

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

6.29.2 Constructor & Destructor Documentation

6.29.2.1 BandwidthManager::BandwidthManager (ServiceLevelAgreement * *sla*, RoundTripTimePinger * *rttp*)

Constructor.

Parameters

sla	ServiceLevelAgreement object.
rttp	RoundTripTimePinger object.

6.29.2.2 BandwidthManager::~BandwidthManager ( )

Destructor.

6.29.3 Member Function Documentation

6.29.3.1 void BandwidthManager::addStream (ManagedStreamInterface * *stream*, const cardinal *sessionID* = 0, const char * *name* = NULL) [virtual]

Add stream to management.

Parameters

stream	Stream to add.
session	SessionID of session to add stream to (0 for no session).
name	Stream name (only for log printing).

Implements QoSManagerInterface.

6.29.3.2 void BandwidthManager::bufferFlushEvent (ManagedStreamInterface * *stream*, const cardinal *layer*) [virtual]

Buffer flush for a given layer.

Parameters

stream	Stream.

Implements QoSManagerInterface.

6.29.3.3 cardinal BandwidthManager::calculateSessionMultiPoints (SessionDescription * *session*, const cardinal *offset*, const cardinal *lastPoint*, ResourceUtilizationMultiPoint * *rumpList*) [private]

6.29.3.4 void **BandwidthManager::doAllocationTrials** (**ResourceUtilizationMultiPoint** * *rumpList,* const **cardinal** *points,* const **card64** *bandwidthLimit* = (**card64**)-1) [private]

6.29.3.5 void **BandwidthManager::doCompleteRemapping** ( ) [private]

6.29.3.6 bool **BandwidthManager::doPartialRemapping** (**StreamDescription** * *streamDescription*) [private]

6.29.3.7 void **BandwidthManager::forceCompleteRemapping** ( ) [inline]

Force a complete remapping.

6.29.3.8 void **BandwidthManager::getFairness** (double & *fairnessSession,* double & *fairnessStream*) const [inline]

Get fairnes settings.

Parameters

fairnessSession	Reference to store session fairness.
fairnessStream	Reference to store stream fairness.

6.29.3.9 void **BandwidthManager::getPartialRemapping** (bool & *enabled,* double & *reservedPortion,* double & *utilizationTolerance,* double & *maxRemappingInterval*) const [inline]

Get partial remapping parameters.

Parameters

enabled	Reference to store true, if partial remappings are enabled; false otherwise.
reservedPortion	Reference to store reserved bandwidth portion (out of [0,1]).
utilizationTolerance	Reference to store utilization tolerance for partial remapping.
maxRemappingInterval	Reference to store maximum interval between two complete remappings.

6.29.3.10 double **BandwidthManager::getPriorityFactor** (const **int8** *streamPriority*) const [inline, private]

6.29.3.11 void BandwidthManager::getQoSOptimizationParameters (cardinal & *maxRUPoints,* double & *utilizationThreshold,* card64 & *bandwidthThreshold,* double & *systemDelayTolerance,* bool & *unlayeredAllocation*) const [inline]

Get QoS optimization parameters.

Parameters

maxRU-Points	Reference to store maximum number of RU points per stream.
utilization-Threshold	Reference to store utilization threshold.
bandwidth-Threshold	Reference to store bandwidth threshold.
system-Delay-Tolerance	Reference to store the system's delay tolerance for the buffering optimization.

6.29.3.12 double BandwidthManager::getResourcePart (const ResourceUtilizationSimplePoint & *rup*) const [inline, private]

6.29.3.13 double BandwidthManager::getResourcePart (const ResourceUtilizationMultiPoint & *rump*) const [inline, private]

6.29.3.14 void BandwidthManager::getRoundTripTimes (StreamDescription * *sd*) [private]

6.29.3.15 double BandwidthManager::getSessionSortingValue (const ResourceUtilizationMultiPoint & *rump*) const [inline, private]

6.29.3.16 double BandwidthManager::getStreamSortingValue (const ResourceUtilizationSimplePoint & *rup*) const [inline, private]

6.29.3.17 void BandwidthManager::intervalChangeEvent (ManagedStreamInterface * *stream,* const bool *isNew,* const card64 *when,* const bool *newRUList*) [virtual]

Interval has changed.

Parameters

stream	Stream with changed interval.
isNew	true, if new interval has been reached; false otherwise.
when	Microseconds to next interval.
newRUList	true, if new resource/utiliztion list has been reached; false otherwise.

Implements QoSManagerInterface.

6.29.3.18 void BandwidthManager::removeStream (ManagedStreamInterface * *stream*) [virtual]

Remove stream from management.

Parameters

stream	Stream to remove.

Implements QoSManagerInterface.

6.29.3.19 void BandwidthManager::reportEvent (ManagedStreamInterface * *stream,* const RTCPReceptionReportBlock * *report,* const cardinal *layer*) [virtual]

Report reception for given layer.

Parameters

stream	Stream.
report	Report.
layer	Layer.

Implements QoSManagerInterface.

6.29.3.20 void BandwidthManager::setFairness (const double *fairnessSession,* const double *fairnessStream*) [inline]

Set fairnes settings.

Parameters

fairness-Session	Session fairness.
fairness-Stream	Stream fairness.

6.29.3.21 void BandwidthManager::setLogStream (std::ostream * *logStream*)

Set log stream.

Parameters

logStream	Stream to write log information to; NULL to disable logging.

6.29.3.22 void BandwidthManager::setPartialRemapping (const bool *enabled,* const double *reservedPortion,* const double *utilizationTolerance,* const double *maxRemappingInterval*) [inline]

Set partial remapping parameters.

Parameters

enabled	true, if partial remappings are enabled; false otherwise.
reserved-Portion	Reserved bandwidth portion (out of [0,1]).
utilization-Tolerance	Utilization tolerance for partial remapping.
max-Remapping-Interval	Maximum interval between two complete remappings.

6.29.3.23 void BandwidthManager::setQoSOptimizationParameters (const cardinal *maxRUPoints,* const double *utilizationThreshold,* const card64 *bandwidthThreshold,* const double *systemDelayTolerance,* const bool *unlayeredAllocation*) [inline]

Set QoS optimization parameters.

Parameters

maxRU-Points	Maximum number of RU points per stream.
utilization-Threshold	Utilization threshold.
bandwidth-Threshold	Bandwidth threshold.
system-Delay-Tolerance	The system's delay tolerance for the buffering optimization.

6.29.3.24 static void BandwidthManager::smoothedUpdate (double & *value,* const double *measured,* const double *alpha*) [inline, static, private]

6.29.3.25 void BandwidthManager::timerEvent ( ) [virtual]

Implementation of TimedThread's timerEvent() method.

See also

TimedThread::timerEvent

Implements TimedThread.

6.29.3.26 bool BandwidthManager::tryAllocation (ResourceUtilizationMultiPoint & *rump,* const card64 *bandwidthLimit* = (card64)-1) [private]

6.29.3.27 void BandwidthManager::updateReservation (StreamDescription * *streamDescription*)

6.29.3.28 void BandwidthManager::updateStream (ManagedStreamInterface * *stream*) [virtual]

Update stream.

Parameters

stream	Stream to be updated.

Implements QoSManagerInterface.

6.29.4 Member Data Documentation

6.29.4.1 double BandwidthManager::AlphaJitter

6.29.4.2 double BandwidthManager::AlphaLossRate

6.29.4.3 card64 BandwidthManager::BandwidthThreshold

6.29.4.4 bool BandwidthManager::Changed [private]

6.29.4.5 card64 BandwidthManager::ClassAvailableBandwidthArray[TrafficClassValues::MaxValues]

6.29.4.6 card64 BandwidthManager::ClassBandwidthArray[TrafficClassValues::MaxValues]

6.29.4.7 cardinal BandwidthManager::CompleteRemappings

6.29.4.8 bool BandwidthManager::EnablePartialRemappings

6.29.4.9 double BandwidthManager::FairnessSession

6.29.4.10 double BandwidthManager::FairnessStream

6.29.4.11 card64 BandwidthManager::LastCompleteRemapping

6.29.4.12 card64 BandwidthManager::LastCompleteRemappingDuration

6.29.4.13 std::ostream* BandwidthManager::Log [private]

6.29.4.14 card64 BandwidthManager::LogStartupTimeStamp [private]

6.29.4.15 card64 BandwidthManager::MaxRemappingInterval

6.29.4.16 cardinal BandwidthManager::MaxRUPoints

6.29.4.17 double BandwidthManager::PartialRemappingPortion

6.29.4.18 cardinal BandwidthManager::PartialRemappings

6.29.4.19 double BandwidthManager::PartialRemappingUtilizationTolerance

6.29.4.20 RoundTripTimePinger* BandwidthManager::RTTP [private]

6.29.4.21 cardinal BandwidthManager::Sessions

6.29.4.22 std::multimap<cardinal,SessionDescription*> BandwidthManager::SessionSet

6.29.4.23 card64 BandwidthManager::SimulatorTime = 0 [static]

6.29.4.24 ServiceLevelAgreement* BandwidthManager::SLA

6.29.4.25 int64 BandwidthManager::SLAUpdateRecommendation[TrafficClassValues::MaxValues]

6.29.4.26 card64 BandwidthManager::StreamIDGenerator

6.29.4.27 cardinal BandwidthManager::Streams

6.29.4.28 std::multimap<ManagedStreamInterface*,StreamDescription*> BandwidthManager::StreamSet

6.29.4.29 double BandwidthManager::SystemDelayTolerance

6.29.4.30 card64 BandwidthManager::TotalAvailableBandwidth

6.29.4.31 card64 BandwidthManager::TotalBandwidth

6.29.4.32 cardinal BandwidthManager::TotalBufferFlushes

6.29.4.33 bool BandwidthManager::UnlayeredAllocation

6.29.4.34 double BandwidthManager::UtilizationThreshold

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

- bandwidthmanager.h

- bandwidthmanager.cc

6.30 RTCPAbstractServer::Client Struct Reference

`#include <rtcpabstractserver.h>`

Public Attributes

- card32 SSRC
- InternetFlow ClientAddress
- card64 TimeStamp
- card64 Timeout
- void * UserData

6.30.1 Detailed Description

Client structure with information on the client.

6.30.2 Member Data Documentation

6.30.2.1 InternetFlow RTCPAbstractServer::Client::ClientAddress

6.30.2.2 card32 RTCPAbstractServer::Client::SSRC

6.30.2.3 card64 RTCPAbstractServer::Client::Timeout

6.30.2.4 card64 RTCPAbstractServer::Client::TimeStamp

6.30.2.5 void* RTCPAbstractServer::Client::UserData

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

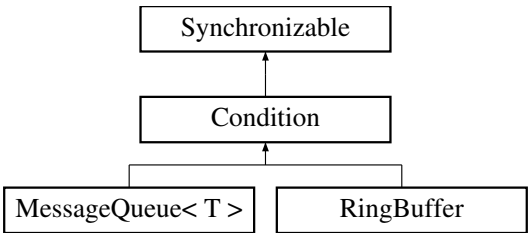
- rtcpabstractserver.h

6.31 Condition Class Reference

Condition.

`#include <condition.h>`

Inheritance diagram for Condition:



Public Member Functions

- Condition (const char *name="Condition", Condition *parentCondition=NULL, const bool recursive=true)
- ~Condition ()
- void signal ()
- void broadcast ()
- bool fired ()
- bool peekFired ()
- void wait ()
- bool timedWait (const card64 microseconds)
- void addParent (Condition *parentCondition)
- void removeParent (Condition *parentCondition)

Private Attributes

- std::set< Condition * > ParentSet
- pthread_cond_t ConditionVariable
- bool Fired
- bool Valid

6.31.1 Detailed Description

Condition.

This class realizes a condition variable.

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

See also

Synchronizable
Thread

6.31.2 Constructor & Destructor Documentation

6.31.2.1 Condition::Condition (const char * *name* = "Condition", Condition * *parentCondition* = NULL, const bool *recursive* = true)

Constructor.

Parameters

name	Name.
parent-Condition	Parent condition.
recursive	true to make condition's mutex recursive; false otherwise (default for Condition!).

6.31.2.2 Condition::~Condition ( )

Destructor.

6.31.3 Member Function Documentation

6.31.3.1 void Condition::addParent (Condition * *parentCondition*)

Add parent condition.

Parameters

parent-Condition	Parent condition to be added.

6.31.3.2 void Condition::broadcast ( )

Broadcast condition: All threads waiting for this variable will be resumed.

6.31.3.3 bool Condition::fired ( ) [inline]

Check, if condition has been fired. This call will reset the fired state.

Returns

true, if condition has been fired; false otherwise.

6.31.3.4 bool Condition::peekFired ( ) [inline]

Check, if condition has been fired. This call will *not* reset the fired state.

Returns

true, if condition has been fired; false otherwise.

6.31.3.5 void Condition::removeParent (Condition * *parentCondition*)

Remove parent condition.

Parameters

parentCondition	Parent condition to be removed.

6.31.3.6 void Condition::signal ( )

Fire condition: One thread waiting for this variable will be resumed.

6.31.3.7 bool Condition::timedWait (const card64 *microseconds*)

Wait for condition with timeout.

Parameters

microseconds	Timeout in microseconds.

Returns

true, if condition has been received; false for timeout.

6.31.3.8 void Condition::wait ( )

Wait for condition without timeout.

6.31.4 Member Data Documentation

6.31.4.1 pthread_cond_t Condition::ConditionVariable [private]

6.31.4.2 bool Condition::Fired [private]

6.31.4.3 std::set<Condition*> Condition::ParentSet [private]

6.31.4.4 bool Condition::Valid [private]

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

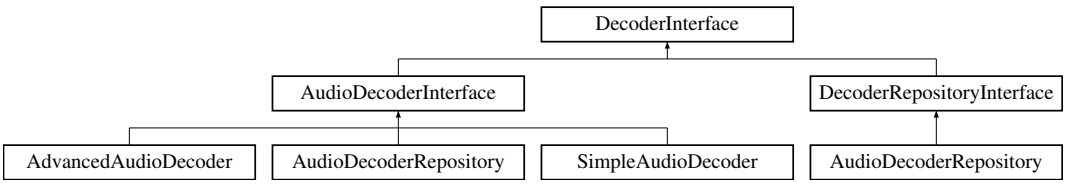
- condition.h
- condition.cc

6.32 DecoderInterface Class Reference

Decoder Interface.

`#include <decoderinterface.h>`

Inheritance diagram for DecoderInterface:



Public Member Functions

- virtual const card16 getTypeID () const =0
- virtual const char * getTypeName () const =0
- virtual void activate ()=0
- virtual void deactivate ()=0
- virtual void reset ()=0
- virtual bool checkNextPacket (DecoderPacket *packet)=0
- virtual void handleNextPacket (const DecoderPacket *decoderPacket)=0
- virtual void getMediaInfo (MediaInfo &mediaInfo) const =0
- virtual card8 getErrorCode () const =0
- virtual card64 getPosition () const =0
- virtual card64 getMaxPosition () const =0

6.32.1 Detailed Description

Decoder Interface.

This class is the interface for a decoder.

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

6.32.2 Member Function Documentation

6.32.2.1 virtual void DecoderInterface::activate ( ) [pure virtual]

Activate the decoder. Usage example: Start an decoder thread.

Implemented in AudioDecoderRepository, AdvancedAudioDecoder, and SimpleAudioDecoder.

6.32.2.2 virtual bool DecoderInterface::checkNextPacket (DecoderPacket * *packet*) [pure virtual]

Check next packet. This function has to set valid packet->Layers and packet->Layer value.

Parameters

decoderPacket	DecoderPacket structure.

Returns

true, if packet is valid; false otherwise.

Implemented in AudioDecoderRepository, AdvancedAudioDecoder, and SimpleAudioDecoder.

6.32.2.3 virtual void DecoderInterface::deactivate ( ) [pure virtual]

Deactivate the decoder. Usage example: Stop an decoder thread.

Implemented in AudioDecoderRepository, AdvancedAudioDecoder, and SimpleAudioDecoder.

6.32.2.4 virtual card8 DecoderInterface::getErrorCode ( ) const [pure virtual]

Get error code Usage example: Return error, if reading from file failed.

Returns

Error code

Implemented in AudioDecoderRepository, AdvancedAudioDecoder, and SimpleAudioDecoder.

6.32.2.5 virtual card64 DecoderInterface::getMaxPosition ( ) const [pure virtual]

Get maximum position in nanoseconds.

Returns

Maximum position in nanoseconds.

Implemented in AudioDecoderRepository, AdvancedAudioDecoder, and SimpleAudioDecoder.

6.32.2.6 virtual void DecoderInterface::getMediaInfo (MediaInfo & *mediaInfo*) const [pure virtual]

Get media info.

Parameters

mediaInfo	Reference to store MediaInfo to.

Implemented in AudioDecoderRepository, AdvancedAudioDecoder, and SimpleAudioDecoder.

6.32.2.7 virtual card64 DecoderInterface::getPosition ( ) const [pure virtual]

Get current position in nanoseconds.

Returns

Position in nanoseconds.

Implemented in AudioDecoderRepository, AdvancedAudioDecoder, and SimpleAudioDecoder.

6.32.2.8 virtual const card16 DecoderInterface::getTypeID ( ) const [pure virtual]

Get the decoder's type ID.

Returns

Decoder's type ID.

Implemented in AudioDecoderRepository, AdvancedAudioDecoder, and SimpleAudioDecoder.

6.32.2.9 virtual const char* DecoderInterface::getTypeName ( ) const [pure virtual]

Get the decoder's name.

Returns

Decoder's name

Implemented in AudioDecoderRepository, AdvancedAudioDecoder, and SimpleAudioDecoder.

6.32.2.10 virtual void DecoderInterface::handleNextPacket (const DecoderPacket * *decoderPacket*) [pure virtual]

Handle next received packet.

Parameters

decoderPacket	DecoderPacket structure.

Implemented in AudioDecoderRepository, AdvancedAudioDecoder, and SimpleAudioDecoder.

6.32.2.11 virtual void DecoderInterface::reset ( ) [pure virtual]

Reset the decoder. Usage example: Reset an decoder thread.

Implemented in AudioDecoderRepository, AdvancedAudioDecoder, and SimpleAudioDecoder.

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

- decoderinterface.h

6.33 DecoderPacket Struct Reference

DecoderPacket.

```
#include <decoderinterface.h>
```

Public Attributes

- void * Buffer
- cardinal Length
- card16 SequenceNumber

- card32 TimeStamp
- card8 PayloadType
- bool Marker
- SourceStateInfo ** SSIArray
- cardinal Layer
- cardinal Layers

6.33.1 Detailed Description

DecoderPacket.

This structure contains packet information for handleNextPacket() call.

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

6.33.2 Member Data Documentation

6.33.2.1 void* DecoderPacket::Buffer

Buffer to write packet payload into.

6.33.2.2 cardinal DecoderPacket::Layer

The packet's layer number, to be set within handleNextPacket(). This is used in RTP-Receiver to decide to which layer the packet's FlowInfo belongs. Set Layer (cardinal)-1, if the packet does not belong to a layer, is invalid etc.

6.33.2.3 cardinal DecoderPacket::Layers

The number of layers of the packet's encoding quality, to be set within handleNext-Packet(). Set to (cardinal)-1, if the packet does not belong to a layer, is invalid etc.

6.33.2.4 cardinal DecoderPacket::Length

Maximum length of payload to be written into Buffer.

6.33.2.5 bool DecoderPacket::Marker

The packet's marker.

6.33.2.6 card8 DecoderPacket::PayloadType

The packet's payload type.

6.33.2.7 card16 DecoderPacket::SequenceNumber

The packet's sequence number.

6.33.2.8 SourceStateInfo** DecoderPacket::SSIArray

Source state info array for packet validation within handleNextPacket().

6.33.2.9 card32 DecoderPacket::TimeStamp

The packet's time stamp.

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

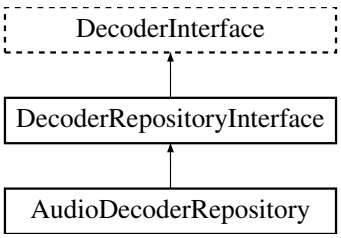
- decoderinterface.h

6.34 DecoderRepositoryInterface Class Reference

Decoder Repository.

`#include <decoderrepositoryinterface.h>`

Inheritance diagram for DecoderRepositoryInterface:



Public Member Functions

- virtual bool selectDecoderForTypeID (const card16 typeID)=0
- virtual DecoderInterface * getCurrentDecoder () const =0

6.34.1 Detailed Description

Decoder Repository.

This class is a repository for decoders.

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

6.34.2 Member Function Documentation

6.34.2.1 virtual DecoderInterface* DecoderRepositoryInterface::getCurrentDecoder ( ) const [pure virtual]

Get DecoderInterface of the current decoder.

Returns

Current decoder's DecoderInterface.

Implemented in AudioDecoderRepository.

6.34.2.2 virtual bool DecoderRepositoryInterface::selectDecoderForTypeID (const card16 *typeID*) [pure virtual]

Select the decoder with the given TypeID to be the current decoder of the repository.

Parameters

typeID	Decoding's type ID.

Returns

true, if decoder for this TypeID was in the repository; false otherwise.

Implemented in AudioDecoderRepository.

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

- decoderrepositoryinterface.h

6.35 DiffServClass Struct Reference

DiffServ Class.

`#include <servicelevelagreement.h>`

Public Attributes

- card64 BytesPerSecond
- double MaxTransferDelay
- double MaxLossRate
- double MaxJitter
- double CostFactor
- double DelayVariability
- card8 TrafficClass

6.35.1 Detailed Description

DiffServ Class.

This is a DiffServ class.

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

6.35.2 Member Data Documentation

6.35.2.1 card64 DiffServClass::BytesPerSecond

Bytes per second.

6.35.2.2 double DiffServClass::CostFactor

Cost factor.

6.35.2.3 double DiffServClass::DelayVariability

Delay variability: Fraction of the transfer delay (out of [0,1]).

6.35.2.4 double DiffServClass::MaxJitter

Maximum jitter.

6.35.2.5 double DiffServClass::MaxLossRate

Maximum loss rate.

6.35.2.6 double DiffServClass::MaxTransferDelay

Maximum transfer delay.

6.35.2.7 card8 DiffServClass::TrafficClass

Traffic class byte.

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

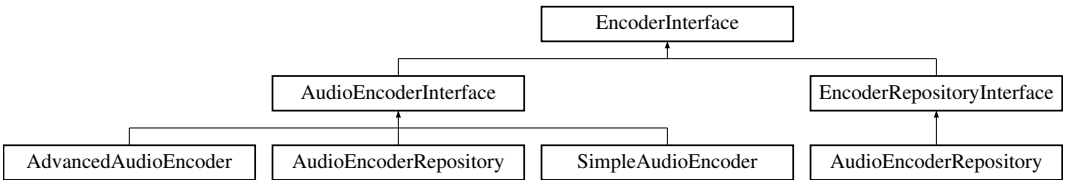
- servicelevelagreement.h

6.36 EncoderInterface Class Reference

Encoder Interface.

`#include <encoderinterface.h>`

Inheritance diagram for EncoderInterface:



Public Member Functions

- virtual const card16 getTypeID () const =0
- virtual const char * getTypeName () const =0
- virtual void activate ()=0
- virtual void deactivate ()=0
- virtual void reset ()=0
- virtual bool checkInterval (card64 &time, bool &newRUList)=0
- virtual bool prepareNextFrame (const cardinal headerSize, const cardinal maxPacketSize, const cardinal flags=0)=0
- virtual cardinal getNextPacket (EncoderPacket *packet)=0
- virtual double getFrameRate () const =0
- virtual AbstractQoSDescription * getQoSDescription (const cardinal pktHeaderSize, const cardinal pktMaxSize, const card64 offset)=0
- virtual void updateQuality (const AbstractQoSDescription *aqd)=0

6.36.1 Detailed Description

Encoder Interface.

This class is the interface for an encoder.

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

6.36.2 Member Function Documentation

6.36.2.1 virtual void EncoderInterface::activate ( ) [pure virtual]

Activate the encoder. Usage example: Start an encoder thread.

Implemented in AudioEncoderRepository, AdvancedAudioEncoder, and SimpleAudioEncoder.

6.36.2.2 virtual bool EncoderInterface::checkInterval (card64 & *time*, bool & *newRUList*) [pure virtual]

Check, when prepareNextFrame() call reaches a new interval.

Parameters

time	Reference to store time to next interval in microseconds.
Reference	to store true, if new resource/utilization list has been reached since last call; false otherwise.

Returns

true, if new interval has been reached since last call; false otherwise.

Implemented in AudioEncoderRepository, AdvancedAudioEncoder, and SimpleAudioEncoder.

6.36.2.3 virtual void EncoderInterface::deactivate ( ) [pure virtual]

Deactivate the encoder. Usage example: Stop an encoder thread.

Implemented in AudioEncoderRepository, AdvancedAudioEncoder, and SimpleAudioEncoder.

6.36.2.4 virtual double EncoderInterface::getFrameRate ( ) const [pure virtual]

Get frame rate.

Returns

Frame rate.

Implemented in AudioEncoderRepository, AdvancedAudioEncoder, and SimpleAudioEncoder.

6.36.2.5 virtual cardinal EncoderInterface::getNextPacket (EncoderPacket * *packet*) [pure virtual]

Get next packet from current frame. The maximum payload length of the packet (the size of packet->Buffer) is in packet->MaxLength.

Parameters

packet	EncoderPacket structure.
buffer	Buffer of the packet to write the data into.
maxLength	Maximum length of the packet

Returns

Real length of the data written into the buffer or 0, if there is no more data of the current frame.

Implemented in AudioEncoderRepository, AdvancedAudioEncoder, and SimpleAudioEncoder.

6.36.2.6 virtual AbstractQoSDescription* EncoderInterface::getQoSDescription (const cardinal *pktHeaderSize,* const cardinal *pktMaxSize,* const card64 *offset*) [pure virtual]

Get QoS description. Important note: This result is a global pointer, it becomes invalid when encoder is deleted!

Parameters

pktHeaderSize	Packet header size.
pktMaxSize	Maximum packet size.
offset	RTP position offset.

Returns

QoS Description.

Implemented in AudioEncoderRepository, AdvancedAudioEncoder, and SimpleAudioEncoder.

6.36.2.7 virtual const card16 EncoderInterface::getTypeID ( ) const [pure virtual]

Get the encoder's type ID.

Returns

Encoder's type ID.

Implemented in AudioEncoderRepository, AdvancedAudioEncoder, and SimpleAudioEncoder.

6.36.2.8 virtual const char* EncoderInterface::getTypeName ( ) const [pure virtual]

Get the encoder's name.

Returns

Encoder's name

Implemented in AudioEncoderRepository, AdvancedAudioEncoder, and SimpleAudioEncoder.

6.36.2.9 virtual bool EncoderInterface::prepareNextFrame (const cardinal *headerSize*, const cardinal *maxPacketSize*, const cardinal *flags* = 0) [pure virtual]

Prepare next frame. Usage example: Read the next frame from file, transform it into packages for transport.

Parameters

headerSize	Size of underlying protocol's header (e.g. RTP packet)
maxPacketSize	Maximum size of packet.
flags	Encoder-specific flags (e.g. compression or encryption).

Returns

true, if there was a next frame; false, if not.

Implemented in AudioEncoderRepository, AdvancedAudioEncoder, and SimpleAudioEncoder.

6.36.2.10 virtual void EncoderInterface::reset ( ) [pure virtual]

Reset the encoder. Usage example: Reset an encoder thread.

Implemented in AudioEncoderRepository, AdvancedAudioEncoder, and SimpleAudioEncoder.

6.36.2.11 virtual void EncoderInterface::updateQuality (const AbstractQoSDescription * *aqd*) [pure virtual]

Update encoder quality to changes made in QoS description returned by getQoSDescription().

See also

EncoderInterface::getQoSDescription

Implemented in AudioEncoderRepository, AdvancedAudioEncoder, and SimpleAudioEncoder.

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

- encoderinterface.h

6.37 EncoderPacket Struct Reference

EncoderPacket.

`#include <encoderinterface.h>`

Public Attributes

- void * Buffer
- cardinal MaxLength
- cardinal Layer
- card8 PayloadType
- bool Marker
- card8 ErrorCode

6.37.1 Detailed Description

EncoderPacket.

This structure contains packet information for getNextPacket() call.

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

6.37.2 Member Data Documentation

6.37.2.1 void* EncoderPacket::Buffer

Buffer to write packet payload into.

6.37.2.2 card8 EncoderPacket::ErrorCode

The packet's error code. If greater than ME_UnrecoverableError, the packet may sent even in case of exceeded traffic shaper buffer!

6.37.2.3 cardinal EncoderPacket::Layer

The packet's layer number, to be set within getNextPacket().

6.37.2.4 bool EncoderPacket::Marker

The packet's marker, to be set within getNextPacket().

6.37.2.5 cardinal EncoderPacket::MaxLength

Maximum length of payload to be written into Buffer.

6.37.2.6 card8 EncoderPacket::PayloadType

The packet's payload type, to be set within getNextPacket().

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

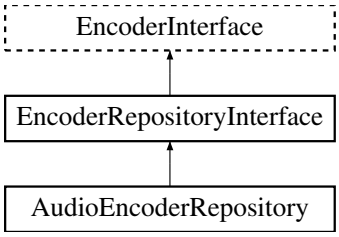
- encoderinterface.h

6.38 EncoderRepositoryInterface Class Reference

Encoder Repository Interface.

`#include <encoderrepositoryinterface.h>`

Inheritance diagram for EncoderRepositoryInterface:



Public Member Functions

- virtual bool selectEncoderForTypeID (const card16 typeID)=0
- virtual EncoderInterface * getCurrentEncoder () const =0

6.38.1 Detailed Description

Encoder Repository Interface.

This class is a repository for encoders.

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

6.38.2 Member Function Documentation

6.38.2.1 virtual EncoderInterface* EncoderRepositoryInterface::getCurrentEncoder ( ) const [pure virtual]

Get EncoderInterface of the current encoder.

Returns

Current encoder's EncoderInterface.

Implemented in AudioEncoderRepository.

6.38.2.2 virtual bool EncoderRepositoryInterface::selectEncoderForTypeID (const card16 *typeID*) [pure virtual]

Select the encoder with the given TypeID to be the current encoder of the repository.

Parameters

typeID	Encoding's type ID.

Returns

true, if encoder for this TypeID was in the repository; false otherwise.

Implemented in AudioEncoderRepository.

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

- encoderrepositoryinterface.h

6.39 FastFourierTransformation Class Reference

Fast Fourier Transformation.

`#include <fft.h>`

Public Member Functions

- FastFourierTransformation (const integer fftlen)
- ~FastFourierTransformation ()
- void fft (int16 *buffer)
- integer * getBitReversed ()

Private Attributes

- integer * BitReversed
- int16 * SinTable
- integer Points
- int16 * A
- int16 * B
- int16 * sptr
- int16 * endptr1
- int16 * endptr2
- integer * br1
- integer * br2
- integer HRplus
- integer HRminus
- integer HIplus
- integer HIminus

6.39.1 Detailed Description

Fast Fourier Transformation.

This class does fast fourier transformation.

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

6.39.2 Constructor & Destructor Documentation

6.39.2.1 FastFourierTransformation::FastFourierTransformation (const integer *fftlen*)

Constructor.

6.39.2.2 FastFourierTransformation::~FastFourierTransformation ( )

Destructor.

6.39.3 Member Function Documentation

6.39.3.1 void FastFourierTransformation::fft (int16 * *buffer*)

Do Fourier transformation.

Parameters

buffer	Input buffer.

6.39.3.2 integer * FastFourierTransformation::getBitReversed ( )

Get BitReversed array.

6.39.4 Member Data Documentation

6.39.4.1 int16* FastFourierTransformation::A [private]

6.39.4.2 int16 * FastFourierTransformation::B [private]

6.39.4.3 integer* FastFourierTransformation::BitReversed [private]

6.39.4.4 integer* FastFourierTransformation::br1 [private]

6.39.4.5 integer * FastFourierTransformation::br2 [private]

6.39.4.6 int16* FastFourierTransformation::endptr1 [private]

6.39.4.7 int16 * FastFourierTransformation::endptr2 [private]

6.39.4.8 integer FastFourierTransformation::HIminus [private]

6.39.4.9 integer FastFourierTransformation::HIplus [private]

6.39.4.10 integer FastFourierTransformation::HRminus [private]

6.39.4.11 integer FastFourierTransformation::HRplus [private]

6.39.4.12 integer FastFourierTransformation::Points [private]

6.39.4.13 int16* FastFourierTransformation::SinTable [private]

6.39.4.14 int16* FastFourierTransformation::sptr [private]

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

- fft.h
- fft.cc

6.40 AdvancedAudioDecoder::FrameFragment Struct Reference

Public Attributes

- cardinal Length
- card16 Fragment
- char Data [0]

6.40.1 Member Data Documentation

6.40.1.1 char AdvancedAudioDecoder::FrameFragment::Data[0]

6.40.1.2 card16 AdvancedAudioDecoder::FrameFragment::Fragment

6.40.1.3 cardinal AdvancedAudioDecoder::FrameFragment::Length

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

- advancedaudiodecoder.h

6.41 AdvancedAudioDecoder::FrameNode Struct Reference

Public Attributes

- card64 Position
- card64 MaxPosition
- cardinal FrameSize
- card16 SamplingRate
- card8 Channels
- card8 Bits
- card8 ErrorCode
- card8 pad
- std::multimap< const card16, FrameFragment * > FragmentSetLL
- std::multimap< const card16, FrameFragment * > FragmentSetRL
- std::multimap< const card16, FrameFragment * > FragmentSetLU
- std::multimap< const card16, FrameFragment * > FragmentSetRU

6.41.1 Member Data Documentation

6.41.1.1 card8 AdvancedAudioDecoder::FrameNode::Bits

6.41.1.2 card8 AdvancedAudioDecoder::FrameNode::Channels

6.41.1.3 card8 AdvancedAudioDecoder::FrameNode::ErrorCode

6.41.1.4 std::multimap<const card16,FrameFragment*> AdvancedAudioDecoder::FrameNode::FragmentSetLL

6.41.1.5 std::multimap<const card16,FrameFragment*> AdvancedAudioDecoder::FrameNode::FragmentSetLU

6.41.1.6 std::multimap<const card16,FrameFragment*> AdvancedAudioDecoder::FrameNode::FragmentSetRL

6.41.1.7 std::multimap<const card16,FrameFragment*> AdvancedAudioDecoder::FrameNode::FragmentSetRU

6.41.1.8 cardinal AdvancedAudioDecoder::FrameNode::FrameSize

6.41.1.9 card64 AdvancedAudioDecoder::FrameNode::MaxPosition

6.41.1.10 card8 AdvancedAudioDecoder::FrameNode::pad

6.41.1.11 card64 AdvancedAudioDecoder::FrameNode::Position

6.41.1.12 card16 AdvancedAudioDecoder::FrameNode::SamplingRate

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

- advancedaudiodecoder.h

6.42 AdvancedAudioDecoder::FrameNodeItem Struct Reference

Public Member Functions

- int operator< (const FrameNodeItem &item) const

Public Attributes

- card64 Position
- FrameNode * Node

6.42.1 Member Function Documentation

6.42.1.1 int AdvancedAudioDecoder::FrameNodeItem::operator< (const FrameNodeItem & *item*) const [inline]

6.42.2 Member Data Documentation

6.42.2.1 FrameNode* AdvancedAudioDecoder::FrameNodeItem::Node

6.42.2.2 card64 AdvancedAudioDecoder::FrameNodeItem::Position

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

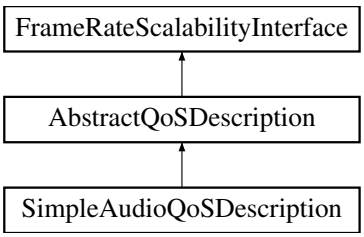
- advancedaudiodecoder.h

6.43 FrameRateScalabilityInterface Class Reference

Frame Rate Scalability Interface.

`#include <frameratescalabilityinterface.h>`

Inheritance diagram for FrameRateScalabilityInterface:



Public Member Functions

- virtual const char * getFrameRateScalabilityClass () const =0
- virtual bool isFrameRateScalable () const =0
- virtual double getMinFrameRate () const =0
- virtual double getMaxFrameRate () const =0
- virtual bool isValidFrameRate (const double frameRate) const =0
- virtual double getNearestValidFrameRate (const double frameRate) const =0
- virtual double getNextFrameRateForRate (const double frameRate) const =0
- virtual double getPrevFrameRateForRate (const double frameRate) const =0
- virtual double getFrameRateScaleFactorForRate (const double frameRate) const =0
- virtual double getFrameRateUtilizationForRate (const double frameRate) const =0
- virtual double getFrameRateUtilizationWeight (const double frameRate) const =0

6.43.1 Detailed Description

Frame Rate Scalability Interface.

This class is an interface for frame rate scalability.

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

6.43.2 Member Function Documentation

6.43.2.1 virtual const char* FrameRateScalabilityInterface::getFrameRateScalabilityClass ( ) const [pure virtual]

Get name of the frame rate scalability class.

Returns

Frame rate scalability class name.

6.43.2.2 virtual double FrameRateScalabilityInterface::getFrameRateScaleFactorForRate (const double *frameRate*) const [pure virtual]

Get scale factor for given frame rate: (rate - MinFrameRate) / (MaxFrameRate - MinFrameRate)

Parameters

frameRate	Frame rate.

Returns

Scale factor (out of [0,1])..

6.43.2.3 virtual double FrameRateScalabilityInterface::getFrameRateUtilizationForRate (const double *frameRate*) const [pure virtual]

Get utilization for given frame rate.

Parameters

frameRate	Frame rate.

Returns

Utilization (out of [0,1]).

6.43.2.4 virtual double FrameRateScalabilityInterface::getFrameRateUtilizationWeight (const double *frameRate*) const [pure virtual]

Get frame rate utilization weight.

Parameters

frameRate	Frame rate.

Returns

Utilization weight.

6.43.2.5 virtual double FrameRateScalabilityInterface::getMaxFrameRate ( ) const [pure virtual]

Get maximum frame rate.

Returns

Maximum frame rate.

6.43.2.6 virtual double FrameRateScalabilityInterface::getMinFrameRate ( ) const [pure virtual]

Get minimum frame rate.

Returns

Minimum frame rate.

6.43.2.7 virtual double FrameRateScalabilityInterface::getNearestValidFrameRate (const double *frameRate*) const [pure virtual]

Get nearest lower valid frame rate for given frame rate.

Parameters

rate	Frame rate.

Returns

Valid frame rate nearest to given rate.

6.43.2.8 virtual double FrameRateScalabilityInterface::getNextFrameRateForRate (const double *frameRate*) const [pure virtual]

Get next higher valid frame rate for given frame rate.

Parameters

frameRate	Frame rate.

Returns

Next higher valid frame rate.

6.43.2.9 virtual double FrameRateScalabilityInterface::getPrevFrameRateForRate (const double *frameRate*) const [pure virtual]

Get next lower valid frame rate for given frame rate.

Parameters

frameRate	Frame rate.

Returns

Next lower valid frame rate.

6.43.2.10 virtual bool FrameRateScalabilityInterface::isFrameRateScalable ( ) const [pure virtual]

Check, if frame rate is scalable.

Returns

true, if frame rate is scalable; false otherwise.

6.43.2.11 virtual bool FrameRateScalabilityInterface::isValidFrameRate (const double *frameRate*) const [pure virtual]

Check, if given frame rate is a valid value.

Parameters

frameRate	Frame rate to be checked.

Returns

true, if given rate is valid; false otherwise.

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

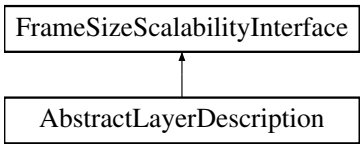
- frameratescalabilityinterface.h

6.44 FrameSizeScalabilityInterface Class Reference

Frame Rate Scalability Interface.

`#include <framesizescalabilityinterface.h>`

Inheritance diagram for FrameSizeScalabilityInterface:



Public Member Functions

- virtual const char * getFrameSizeScalabilityClass () const =0
- virtual bool isFrameSizeScalable () const =0
- virtual bool isVariableBitrate () const =0
- virtual cardinal getMinPayloadFrameSizeForDelay (const double frameRate, const cardinal bufferDelay) const =0
- virtual cardinal getMaxPayloadFrameSizeForDelay (const double frameRate, const cardinal bufferDelay) const =0
- virtual cardinal getMaxFrameCountForDelay (const double frameRate, const cardinal bufferDelay) const =0
- virtual bool isValidPayloadFrameSize (const double frameRate, const cardinal bufferDelay, const cardinal frameSize) const =0
- virtual cardinal getNearestValidPayloadFrameSize (const double frameRate, const cardinal bufferDelay, const cardinal frameSize) const =0
- virtual cardinal getNextPayloadFrameSizeForDelayAndSize (const double frameRate, const cardinal bufferDelay, const cardinal frameSize) const =0
- virtual cardinal getPrevPayloadFrameSizeForDelayAndSize (const double frameRate, const cardinal bufferDelay, const cardinal frameSize) const =0
- virtual double getPayloadFrameSizeScaleFactorForDelayAndSize (const double frameRate, const cardinal bufferDelay, const cardinal frameSize) const =0
- virtual double getPayloadFrameSizeUtilizationForDelayAndSize (const double frameRate, const cardinal bufferDelay, const cardinal frameSize) const =0
- virtual double getFrameSizeUtilizationWeight (const double frameRate) const =0
- virtual cardinal getMaxBufferDelay (const double frameRate) const =0
- virtual cardinal getNextBufferDelayForDelay (const double frameRate, const cardinal bufferDelay) const =0
- virtual cardinal getPrevBufferDelayForDelay (const double frameRate, const cardinal bufferDelay) const =0

6.44.1 Detailed Description

Frame Rate Scalability Interface.

This class is an interface for frame size scalability. Important node: All frames sizes in this class are payload frame sizes!

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

6.44.2 Member Function Documentation

6.44.2.1 virtual const char* FrameSizeScalabilityInterface::getFrameSizeScalabilityClass ( ) const [pure virtual]

Get name of the frame size scalability class.

Returns

Frame size scalability class name.

6.44.2.2 virtual double FrameSizeScalabilityInterface::getFrameSizeUtilizationWeight (const double *frameRate*) const [pure virtual]

Get frame size utilization weight.

Parameters

frameRate	Frame rate.

Returns

Utilization weight.

6.44.2.3 virtual cardinal FrameSizeScalabilityInterface::getMaxBufferDelay (const double *frameRate*) const [pure virtual]

Get maximum buffer delay. The *minimum* buffer delay is always 1.

Parameters

frameRate	Frame rate.

Returns

Maximum buffer delay.

6.44.2.4 virtual cardinal FrameSizeScalabilityInterface::getMaxFrameCountForDelay (const double *frameRate,* const cardinal *bufferDelay*) const [pure virtual]

Get maximum number of frames for given buffer delay (in frame rate units).

Parameters

frameRate	Frame rate.
bufferDelay	Buffer delay in frame rate units.

Returns

Maximum number of frames.

6.44.2.5 virtual cardinal FrameSizeScalabilityInterface::getMaxPayloadFrameSizeForDelay (const double *frameRate,* const cardinal *bufferDelay*) const [pure virtual]

Get maximum payload frame size for given buffer delay (in frame rate units).

Parameters

frameRate	Frame rate.
bufferDelay	Buffer delay in frame rate units.

Returns

Maximum payload frame size.

6.44.2.6 virtual cardinal FrameSizeScalabilityInterface::getMinPayloadFrameSizeForDelay (const double *frameRate,* const cardinal *bufferDelay*) const [pure virtual]

Get minimum payload frame size for given buffer delay (in frame rate units).

Parameters

frameRate	Frame rate.
bufferDelay	Buffer delay in frame rate units.

Returns

Minimum payload frame size.

6.44.2.7 virtual cardinal FrameSizeScalabilityInterface::getNearestValidPayloadFrameSize (const double *frameRate,* const cardinal *bufferDelay,* const cardinal *frameSize*) const [pure virtual]

Get nearest lower valid payload frame rate for given frame rate for given buffer delay (in frame rate units).

Parameters

frameRate	Frame rate.
bufferDelay	Buffer delay in frame rate units.
frameSize	Payload frame size.

Returns

Valid payload frame size nearest to given size for given buffer delay.

6.44.2.8 virtual cardinal FrameSizeScalabilityInterface::getNextBufferDelayForDelay (const double *frameRate,* const cardinal *bufferDelay*) const [pure virtual]

Get next higher valid buffer delay for given buffer delay.

Parameters

frameRate	Frame rate.
bufferDelay	Buffer delay in frame rate units.

Returns

Next higher valid buffer delay.

6.44.2.9 virtual cardinal FrameSizeScalabilityInterface::getNextPayloadFrameSizeForDelayAndSize (const double *frameRate,* const cardinal *bufferDelay,* const cardinal *frameSize*) const [pure virtual]

Get next higher valid payload frame size for given buffer delay (in frame rate units) and payload frame size.

Parameters

frameRate	Frame rate.
bufferDelay	Buffer delay in frame rate units.
frameSize	Payload frame size.

Returns

Next higher valid payload frame size for given buffer delay.

6.44.2.10 virtual double FrameSizeScalabilityInterface::getPayloadFrameSizeScaleFactorForDelayAndSize (const double *frameRate,* const cardinal *bufferDelay,* const cardinal *frameSize*) const [pure virtual]

Get scale factor for given buffer delay (in frame rate units) and payload frame size: (rate - MinFrameSize) / (MaxFrameRate - MinFrameSize)

Parameters

frameRate	Frame rate.
frameSize	Frame size.

Returns

Scale factor (out of [0,1])..

6.44.2.11 virtual double FrameSizeScalabilityInterface::getPayloadFrameSizeUtilizationForDelayAndSize (const double *frameRate,* const cardinal *bufferDelay,* const cardinal *frameSize*) const [pure virtual]

Get utilization for given buffer delay (in frame rate units) and payload frame size.

Parameters

frameRate	Frame rate.
frameSize	Payload frame size.

Returns

Utilization (out of [0,1])..

6.44.2.12 virtual cardinal FrameSizeScalabilityInterface::getPrevBufferDelayForDelay (const double *frameRate,* const cardinal *bufferDelay*) const [pure virtual]

Get next lower valid buffer delay for given buffer delay.

Parameters

frameRate	Frame rate.
bufferDelay	Buffer delay in frame rate units.

Returns

Next lower valid buffer delay.

6.44.2.13 virtual cardinal FrameSizeScalabilityInterface::getPrevPayloadFrameSizeForDelayAndSize (const double *frameRate,* const cardinal *bufferDelay,* const cardinal *frameSize*) const [pure virtual]

Get next lower valid payload frame size for given buffer delay (in frame rate units) and frame size.

Parameters

frameRate	Frame rate.
bufferDelay	Buffer delay in frame rate units.
frameSize	Payload frame size.

Returns

Next lower valid payload frame size for given buffer delay.

6.44.2.14 virtual bool FrameSizeScalabilityInterface::isFrameSizeScalable ( ) const [pure virtual]

Check, if frame size is scalable.

Returns

true, if frame size is scalable; false otherwise.

6.44.2.15 virtual bool FrameSizeScalabilityInterface::isValidPayloadFrameSize (const double *frameRate,* const cardinal *bufferDelay,* const cardinal *frameSize*) const [pure virtual]

Check, if given payload frame size is a valid value for given buffer delay (in frame rate units).

Parameters

frameRate	Frame rate.
bufferDelay	Buffer delay in frame rate units.
frameSize	Payload frame size to be checked.

Returns

true, if given size is valid; false otherwise.

6.44.2.16 virtual bool FrameSizeScalabilityInterface::isVariableBitrate ( ) const [pure virtual]

Check, if frame size is variable bitrate (frame sizes are different for each frame; the frame size given is the frame size necessary to be reserved for a given buffer delay).

Returns

true, if frame size is variable bitrate; false otherwise.

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

- framesizescalabilityinterface.h

6.45 icmp_filter Struct Reference

Public Attributes

- card32 data

6.45.1 Member Data Documentation

6.45.1.1 card32 icmp_filter::data

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

- roundtriptimepinger.cc

6.46 in6_flowlabel_req Struct Reference

```
#include <tdin6.h>
```

Public Attributes

- struct in6_addr flr_dst
- __u32 flr_label
- __u8 flr_action
- __u8 flr_share
- __u16 flr_flags
- __u16 flr_expires
- __u16 flr_linger
- __u32 __flr_pad

6.46.1 Detailed Description

IMPORTANT NOTE: This is an extraction of <linux/in6.h>, which cannot be included due to incompatibilities! This file will be replaced, when incompatibilities are solved!

6.46.2 Member Data Documentation

6.46.2.1 __u32 in6_flowlabel_req::__flr_pad

6.46.2.2 __u8 in6_flowlabel_req::flr_action

6.46.2.3 struct in6_addr in6_flowlabel_req::flr_dst

6.46.2.4 __u16 in6_flowlabel_req::flr_expires

6.46.2.5 __u16 in6_flowlabel_req::flr_flags

6.46.2.6 __u32 in6_flowlabel_req::flr_label

6.46.2.7 __u16 in6_flowlabel_req::flr_linger

6.46.2.8 __u8 in6_flowlabel_req::flr_share

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

- tdin6.h

6.47 InfoEntry Struct Reference

InfoEntry.

`#include <qinfotabwidget.h>`

Public Attributes

- const char * ID
- const char * Title
- const char * Help

6.47.1 Detailed Description

InfoEntry.

This structure describes a single info string for QInfoWidget and QInfoTabWidget.

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

6.47.2 Member Data Documentation

6.47.2.1 const char* InfoEntry::Help

Help text for the info string.

6.47.2.2 const char* InfoEntry::ID

ID for update() call.

See also

QInfoWidget::update.
QInfoTabWidget::update

6.47.2.3 const char* InfoEntry::Title

Title of the info string.

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

- qinfotabwidget.h

6.48 InfoTable Struct Reference

InfoTable.

```
#include <qinfotabwidget.h>
```

Public Attributes

- const cardinal Entries
- const InfoEntry * Entry

6.48.1 Detailed Description

InfoTable.

This structure is a table of InfoEntries for QInfoWidget and QInfoTabWidget.

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

6.48.2 Member Data Documentation

6.48.2.1 const cardinal InfoTable::Entries

Number of entries in the table.

6.48.2.2 const InfoEntry* InfoTable::Entry

List of InfoEntries.

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

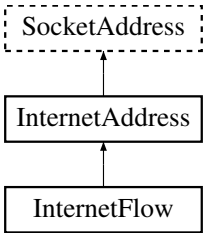
- qinfotabwidget.h

6.49 InternetAddress Class Reference

Socket Address.

`#include <internetaddress.h>`

Inheritance diagram for InternetAddress:



Public Member Functions

- InternetAddress ()

- InternetAddress (const InternetAddress &address)
- InternetAddress (const String &address)
- InternetAddress (const String &hostName, const card16 port)
- InternetAddress (const card16 port)
- InternetAddress (const sockaddr *address, const socklen_t length)
- ~InternetAddress ()
- void reset ()
- SocketAddress * duplicate () const
- void init (const InternetAddress &address)
- void init (const String &hostName, const card16 port)
- void init (const card16 port)
- void init (const sockaddr *address, const socklen_t length)
- InternetAddress & operator= (const InternetAddress &source)
- card16 getPort () const
- void setPort (const card16 port)
- bool isValid () const
- integer getFamily () const
- String getAddressString (const cardinal format=PF_Default) const
- bool isIPv4 () const
- bool isIPv4compatible () const
- bool isIPv6 () const
- bool isNull () const
- bool isUnspecified () const
- bool isLoopback () const
- bool isLinkLocal () const
- bool isSiteLocal () const
- bool isGlobal () const
- bool isMulticast () const
- bool isBroadcast () const
- bool isUnicast () const
- bool isReserved () const
- bool isNodeLocalMulticast () const
- bool isLinkLocalMulticast () const
- bool isSiteLocalMulticast () const
- bool isOrgLocalMulticast () const
- bool isGlobalMulticast () const
- cardinal getSystemAddress (sockaddr *buffer, const socklen_t length, const cardinal type) const
- bool setSystemAddress (const sockaddr *address, const socklen_t length)
- int operator== (const InternetAddress &address) const
- int operator!= (const InternetAddress &address) const
- int operator< (const InternetAddress &address) const
- int operator<= (const InternetAddress &address) const
- int operator> (const InternetAddress &address) const
- int operator>= (const InternetAddress &address) const
- PortableAddress getPortableAddress () const
- InternetAddress (const PortableAddress &address)
- void init (const PortableAddress &address)

Static Public Member Functions

- static cardinal getHostByName (const String &name, card16 *myadr, card16 *myscope=NULL)
- static card16 getServiceByName (const char *name)
- static bool getFullHostName (char *str, const size_t size)
- static bool hasIPv6 ()
- static InternetAddress getLocalAddress (const InternetAddress &peer)
- static bool setIPv4Address (const InternetAddress &address, in_addr *ipv4-Address)
- static InternetAddress getIPv4Address (const in_addr &ipv4Address)
- static card32 calculateChecksum (card8 *buffer, const cardinal bytes, card32 sum)
- static card32 wrapChecksum (card32 sum)

Static Public Attributes

- static bool UseIPv6 = checkIPv6()

Static Private Member Functions

- static bool checkIPv6 ()

Private Attributes

- union {
  card16 Host16 [8]
  card32 Host32 [4]
  in6_addr Address
  } AddrSpec

- card16 Port
- card16 ScopeID
- bool Valid

6.49.1 Detailed Description

Socket Address.

This class manages an internet address.

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

6.49.2 Constructor & Destructor Documentation

6.49.2.1 InternetAddress::InternetAddress ( )

Constructor for an empty internet address.

6.49.2.2 InternetAddress::InternetAddress (const InternetAddress & *address*)

Constructor for an internet address from an internet address.

Parameters

address	Internet address.

6.49.2.3 InternetAddress::InternetAddress (const String & *address*)

Constructor for a internet address given by a string. Examples: "gaffel:7500", "12.34.-56.78:7500", "3ffe:4711::0!7500", "odin:7500", "ipv6-odin:7500".

Parameters

address	Address string.

6.49.2.4 InternetAddress::InternetAddress (const String & *hostName,* const card16 *port*)

Constructor for a internet address given by host name and port.

Parameters

hostName	Host name.
port	Port number.

6.49.2.5 InternetAddress::InternetAddress (const card16 *port*)

Constructor for INADDR_ANY address with given port.

Parameters

port	Port number.

6.49.2.6 InternetAddress::InternetAddress (const sockaddr * *address*, const socklen_t *length*)

Constructor for a internet address from the system's sockaddr structure. The sockaddr structure may be sockaddr_in (IPv4) or sockaddr_in6 (IPv6).

Parameters

address	sockaddr.
length	Length of sockaddr (sizeof(sockaddr_in) or sizeof(sockaddr_in6)).

6.49.2.7 InternetAddress::~InternetAddress ( )

Destructor.

6.49.2.8 InternetAddress::InternetAddress (const PortableAddress & *address*)

Constructor for InternetAddress from PortableAddress.

Parameters

address	PortableAddress.

6.49.3 Member Function Documentation

6.49.3.1 card32 InternetAddress::calculateChecksum (card8 * *buffer*, const cardinal *bytes*, card32 *sum*) [static]

Calculate internet checksum.

Parameters

buffer	Buffer to calculate checksum from.
bytes	Number of bytes.
sum	Checksum value to add.

Returns

Checksum.

6.49.3.2 bool InternetAddress::checkIPv6 ( ) [static, private]

6.49.3.3 SocketAddress * InternetAddress::duplicate ( ) const [virtual]

duplicate() implementation of SocketAddress.

See also

SocketAddress::duplicate

Implements SocketAddress.

Reimplemented in InternetFlow.

6.49.3.4 String InternetAddress::getAddressString (const cardinal *format* = PF_Default) const [virtual]

getAddressString() implementation of SocketAddress.

See also

SocketAddress::getAddressString

Implements SocketAddress.

Reimplemented in InternetFlow.

6.49.3.5 integer InternetAddress::getFamily ( ) const [virtual]

getFamily() implementation of SocketAddress.

See also

SocketAddress::getFamily

Implements SocketAddress.

6.49.3.6 bool InternetAddress::getFullHostName (char * *str*, const size_t *size*) [static]

Get the computer's full locat name (format: name.domain).

Parameters

str	Buffer to store name to.
size	Size of buffer.

Returns

true for success; false otherwise.

6.49.3.7 cardinal InternetAddress::getHostByName (const String & *name,* card16 * *myadr,* card16 * *myscope* = NULL) [static]

Wrapper for system's gethostbyname() function. This version does support IPv6 addresses even if the system itself does not support IPv6. IPv6 addresses are then converted to IPv4 if possible (IPv4-mapped IPv6).

Parameters

name	Host name.
myadr	Storage space to save a IPv6 address (16 bytes).
myscope	Storage space to save IPv6 link local scope ID to (or NULL).
length	Length of the address saved in myaddr or 0 in case of failure.

6.49.3.8 InternetAddress InternetAddress::getIPv4Address (const in_addr & *ipv4Address*) [static]

Get IPv4 InternetAddress from in_addr structure.

Parameters

ipv4Address	in_addr to get address from.

Returns

InternetAddress.

6.49.3.9 InternetAddress InternetAddress::getLocalAddress (const InternetAddress & *peer*) [static]

Get the local host address. The parameter peer gives the address of the other host.

Parameters

peer	Address of peer.

Returns

Local internet address.

Examples: localhost => localhost address (127.0.0.1 or ::1). ethernet-host => ethernet interface address. internet-address => dynamic-ip address set by pppd.

6.49.3.10 card16 InternetAddress::getPort ( ) const [virtual]

getPort() implementation of SocketAddress.

See also

SocketAddress::getPort

Implements SocketAddress.

6.49.3.11 PortableAddress InternetAddress::getPortableAddress ( ) const [inline]

Get PortableAddress from InternetAddress.

Returns

PortableAddress.

6.49.3.12 card16 InternetAddress::getServiceByName (const char * *name*) [static]

Get port number for given service (e.g. http).

Parameters

name	Service name (e.g. "http" or "telnet").

Returns

Port number of 0 if unknown.

6.49.3.13 cardinal InternetAddress::getSystemAddress (sockaddr * *buffer*, const socklen_t *length*, const cardinal *type*) const [virtual]

getSystemAddress() implementation of SocketAddress

See also

SocketAddress::getSystemAddress

Implements SocketAddress.

Reimplemented in InternetFlow.

6.49.3.14 static bool InternetAddress::hasIPv6 ( ) [inline, static]

Check, if IPv6 support is available.

Returns

true, if IPv6 support is available; false otherwise.

6.49.3.15 void InternetAddress::init (const InternetAddress & *address*)

Initialize internet address from internet address.

6.49.3.16 void InternetAddress::init (const String & *hostName,* const card16 *port*)

Initialize internet address with given host name and port.

Parameters

hostName	Host name.
port	Port number.

6.49.3.17 void InternetAddress::init (const card16 *port*)

Initialize internet address with INADDR_ANY and given port.

Parameters

port	Port number.

6.49.3.18 void InternetAddress::init (const sockaddr * *address,* const socklen_t *length*)

Initialize internet address from the system's sockaddr structure. The sockaddr structure may be sockaddr_in (IPv4) or sockaddr_in6 (IPv6).

Parameters

address	sockaddr.
length	Length of sockaddr (sizeof(sockaddr_in) or sizeof(sockaddr_in6)).

6.49.3.19 void InternetAddress::init (const PortableAddress & *address*)

Initialize InternetAddress from PortableAddress.

Parameters

address	PortableAddress.

6.49.3.20 bool InternetAddress::isBroadcast ( ) const [inline]

Check, if internet address broadcast address.

Returns

true or false.

6.49.3.21 bool InternetAddress::isGlobal ( ) const [inline]

Check, if internet address is global, that is !(isLinkLocal() || isSiteLocal()).

Returns

true or false.

6.49.3.22 bool InternetAddress::isGlobalMulticast ( ) const [inline]

Check, if internet address is a global IPv6 multicast address.

Returns

true or false.

6.49.3.23 bool InternetAddress::isIPv4 ( ) const [inline]

Check, if internet address is an IPv4 or IPv4-mapped address (a.b.c.d or ::ffff:a.b.c.d).

Returns

true, if address is an IPv4 or IPv4-mapped address; false otherwise.

6.49.3.24 bool InternetAddress::isIPv4compatible ( ) const [inline]

Check, if internet address is an IPv4-compatible IPv6 address (::a.b.c.d and NOT ::ffff-:a.b.c.d).

Returns

true, if address is an IPv4-compatible IPv6 address; false otherwise.

6.49.3.25 bool InternetAddress::isIPv6 ( ) const [inline]

Check, if internet address is a real (not IPv4-mapped) IPv6 address. Addresses which return true here can be used with labeled flows by class Socket.

Returns

true, if address is real IPv6; false otherwise.

6.49.3.26 bool InternetAddress::isLinkLocal ( ) const [inline]

Check, if internet address is link local (IPv6) or 127.x.y.z (IPv4).

Returns

true or false.

6.49.3.27 bool InternetAddress::isLinkLocalMulticast ( ) const [inline]

Check, if internet address is a link local IPv6 multicast address.

Returns

true or false.

6.49.3.28 bool InternetAddress::isLoopback ( ) const [inline]

Check, if the address is loopback address (127.x.y.z for IPv4 or ::1 for IPv6).

Returns

true, if the address is loopback address; false otherwise.

6.49.3.29 bool InternetAddress::isMulticast ( ) const [inline]

Check, if internet address is a multicast address.

Returns

true or false.

6.49.3.30 bool InternetAddress::isNodeLocalMulticast ( ) const [inline]

Check, if internet address is a node local IPv6 multicast address.

Returns

true or false.

6.49.3.31 bool InternetAddress::isNull ( ) const [inline]

Check, if the address is null address (0.0.0.0 for IPv4 or :: for IPv6) and port number 0. To skip port number check, use isUnspecified().

Returns

true, if the address is null; false otherwise.

See also

isUnspecified

6.49.3.32 bool InternetAddress::isOrgLocalMulticast ( ) const [inline]

Check, if internet address is a organization local IPv6 multicast address.

Returns

true or false.

6.49.3.33 bool InternetAddress::isReserved ( ) const [inline]

Check, if internet address is an reserved address.

Returns

true or false.

6.49.3.34 bool InternetAddress::isSiteLocal ( ) const [inline]

Check, if internet address is site local (IPv6) or 127.x.y.z, 192.168.x.y or 10.x.y.z or within {172.16.0.0 to 127.31.255.255} (IPv4).

Returns

true or false.

6.49.3.35 bool InternetAddress::isSiteLocalMulticast ( ) const [inline]

Check, if internet address is a site local IPv6 multicast address.

Returns

true or false.

6.49.3.36 bool InternetAddress::isUnicast ( ) const [inline]

Check, if internet address is an unicast address (not broadcast or multicast).

Returns

true or false.

6.49.3.37 bool InternetAddress::isUnspecified ( ) const [inline]

Check, if the address is null address (0.0.0.0 for IPv4 or :: for IPv6). This function does not check the port number. To also check the port number, use isNull().

Returns

true, if the address is null; false otherwise.

See also

isNull

6.49.3.38 bool InternetAddress::isValid ( ) const [virtual]

isValid() implementation of SocketAddress.

See also

SocketAddress::isValid

Implements SocketAddress.

6.49.3.39 int InternetAddress::operator!= (const InternetAddress & *address*) const [inline]

Implementation of != operator.

6.49.3.40 int InternetAddress::operator< (const InternetAddress & *address*) const

Implementation of < operator.

6.49.3.41 int InternetAddress::operator<= (const InternetAddress & *address*) const [inline]

Implementation of <= operator.

6.49.3.42 InternetAddress& InternetAddress::operator= (const InternetAddress & *source*) [inline]

Implementation of = operator.

6.49.3.43 int InternetAddress::operator== (const InternetAddress & *address*) const

Implementation of == operator.

6.49.3.44 int InternetAddress::operator> (const InternetAddress & *address*) const

Implementation of > operator.

6.49.3.45 int InternetAddress::operator>= (const InternetAddress & *address*) const [inline]

Implementation of >= operator.

6.49.3.46 void InternetAddress::reset ( ) [virtual]

Reset internet address.

Implements SocketAddress.

Reimplemented in InternetFlow.

6.49.3.47 bool InternetAddress::setIPv4Address (const InternetAddress & *address*, in_addr * *ipv4Address*) [static]

Set in_addr structure from InternetAddress (IPv4 only!).

Parameters

address	InternetAddress.
ipv4Address	Pointer to in_addr to write address to.

Returns

true for success; false otherwise (IPv6 address).

6.49.3.48 void InternetAddress::setPort (const card16 *port*) [virtual]

setPort() implementation of SocketAddress.

See also

SocketAddress::setPort

Implements SocketAddress.

6.49.3.49 bool InternetAddress::setSystemAddress (const sockaddr * *address*, const socklen_t *length*) [virtual]

setSystemAddress() implementation of SocketAddress.

See also

SocketAddress::setSystemAddress

Implements SocketAddress.

6.49.3.50 card32 InternetAddress::wrapChecksum (card32 *sum*) [static]

Prepare checksum for writing into header: Wrap sum and convert byte order.

Parameters

sum	Checksum.

Returns

Checksum.

6.49.4 Member Data Documentation

6.49.4.1 in6_addr InternetAddress::Address

6.49.4.2 union { ... } InternetAddress::AddrSpec [private]

Host address in network byte order. IPv4 addresses are converted to IPv4-mapped IPv6 addresses.

6.49.4.3 card16 InternetAddress::Host16[8]

6.49.4.4 card32 InternetAddress::Host32[4]

6.49.4.5 card16 InternetAddress::Port [private]

Port number.

6.49.4.6 card16 InternetAddress::ScopeID [private]

Scope ID.

6.49.4.7 bool InternetAddress::UseIPv6 = checkIPv6() [static]

Static variable which shows the availability of IPv6. Setting this variable to false on an IPv6 system simulates an IPv4-only system.

Do ∗not∗ change this variable if Socket or InternetAddress objects are already in use!!!

6.49.4.8 bool InternetAddress::Valid [private]

Is address valid?

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

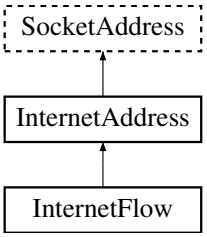
- internetaddress.h
- internetaddress.cc

6.50 InternetFlow Class Reference

Internet Flow.

`#include <internetflow.h>`

Inheritance diagram for InternetFlow:



Public Member Functions

- InternetFlow ()
- InternetFlow (const InternetFlow &flow)
- InternetFlow (const InternetAddress &address, const card32 flowLabel, const card8 trafficClass)
- void reset ()
- SocketAddress ∗ duplicate () const
- String getAddressString (const cardinal format=PF_Default) const

- cardinal getSystemAddress (sockaddr *buffer, const socklen_t length, const cardinal type) const
- bool setSystemAddress (sockaddr *address, socklen_t length)
- card32 getFlowInfo () const
- card32 getFlowLabel () const
- void setFlowLabel (const card32 flowLabel)
- card8 getTrafficClass () const
- void setTrafficClass (const card8 trafficClass)

Private Attributes

- card32 FlowInfo

6.50.1 Detailed Description

Internet Flow.

This class inherits InternetAddress and contains an additional flow label for IPv6 support.

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

6.50.2 Constructor & Destructor Documentation

6.50.2.1 InternetFlow::InternetFlow ( )

Constructor for a new InternetFlow.

6.50.2.2 InternetFlow::InternetFlow (const InternetFlow & *flow*)

Constructor for a new InternetFlow.

Parameters

flow	InternetFlow to be copied.

6.50.2.3 InternetFlow::InternetFlow (const InternetAddress & *address*, const card32 *flowLabel*, const card8 *trafficClass*)

Constructor for a new InternetFlow.

Parameters

address	InternetAddress.
flowLabel	Flow label (20 bits).
trafficClass	Traffic class (8 bits).

6.50.3 Member Function Documentation

6.50.3.1 SocketAddress * InternetFlow::duplicate ( ) const [virtual]

duplicate() implementation of SocketAddress.

See also

SocketAddress::duplicate

Reimplemented from InternetAddress.

6.50.3.2 String InternetFlow::getAddressString (const cardinal *format* = PF_Default) const [virtual]

getAddressString() implementation of SocketAddress.

See also

SocketAddress::getAddressString

Reimplemented from InternetAddress.

6.50.3.3 card32 InternetFlow::getFlowInfo ( ) const [inline]

Get IPv6 flow info: (flowLabel | (trafficClass << 20)).

Returns

Flow info.

6.50.3.4 card32 InternetFlow::getFlowLabel ( ) const [inline]

Get flow label.

Returns

Flow label.

6.50.3.5 cardinal InternetFlow::getSystemAddress (sockaddr * *buffer,* const socklen_t *length,* const cardinal *type*) const [virtual]

getSystemAddress() implementation of SocketAddressInterface.

See also

SocketAddressInterface::getSystemAddress

Reimplemented from InternetAddress.

6.50.3.6 card8 InternetFlow::getTrafficClass ( ) const [inline]

Get traffic class.

Returns

Traffic class.

6.50.3.7 void InternetFlow::reset ( ) [virtual]

Reset flow info.

Reimplemented from InternetAddress.

6.50.3.8 void InternetFlow::setFlowLabel (const card32 *flowLabel*) [inline]

Set flow label.

Parameters

flowLabel	Flow label.

6.50.3.9 bool InternetFlow::setSystemAddress (sockaddr * *address,* socklen_t *length*)

setSystemAddress() implementation of SocketAddressInterface.

See also

SocketAddressInterface::setSystemAddress

6.50.3.10 void InternetFlow::setTrafficClass (const card8 *trafficClass*) [inline]

Set traffic class.

Parameters

trafficClass	New traffic class.

6.50.4 Member Data Documentation

6.50.4.1 card32 InternetFlow::FlowInfo [private]

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

- internetflow.h
- internetflow.cc

6.51 Layer Struct Reference

Public Attributes

- card64 BytesPerSecond
- card32 PacketsPerSecond
- card32 FrameSize

6.51.1 Member Data Documentation

6.51.1.1 card64 Layer::BytesPerSecond

6.51.1.2 card32 Layer::FrameSize

6.51.1.3 card32 Layer::PacketsPerSecond

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

- advancedaudiopacket.cc

6.52 LayerClassMapping Struct Reference

Layer Class Mapping.

```
#include <resourceutilizationpoint.h>
```

Public Attributes

- cardinal Possibilities
- LayerClassMappingPossibility Possibility [TrafficClassValues::MaxValues]

6.52.1 Detailed Description

Layer Class Mapping.

This structure contains a list of possible layer to DiffServ class mapping.

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

6.52.2 Member Data Documentation

6.52.2.1 cardinal LayerClassMapping::Possibilities

Number of possible DiffServ classes for the layer.

6.52.2.2 LayerClassMappingPossibility LayerClassMapping::Possibility[TrafficClassValues::MaxValues]

List of possible DiffServ classes for the layer.

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

- resourceutilizationpoint.h

6.53 LayerClassMappingPossibility Struct Reference

Layer Class Mapping Possibility.

`#include <resourceutilizationpoint.h>`

Public Attributes

- cardinal Class
- cardinal BufferDelay
- double Cost
- card64 Bandwidth

6.53.1 Detailed Description

Layer Class Mapping Possibility.

This structure contains a possible layer to DiffServ class mapping.

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

6.53.2 Member Data Documentation

6.53.2.1 card64 LayerClassMappingPossibility::Bandwidth

Bandwidth.

6.53.2.2 cardinal LayerClassMappingPossibility::BufferDelay

Buffer delay.

6.53.2.3 cardinal LayerClassMappingPossibility::Class

Class ID.

6.53.2.4 double LayerClassMappingPossibility::Cost

Cost.

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

- resourceutilizationpoint.h

6.54 Level Struct Reference

Public Attributes

- card32 BytesPerSecondScale
- card32 PacketsPerSecondScale
- card32 FramesPerSecond
- card32 FramesPerSecondScale
- card8 QualityLayers
- Layer QualityLayer [MaxQualityLayers]

Static Public Attributes

- static const cardinal MaxQualityLayers = 4

6.54.1 Member Data Documentation

6.54.1.1 card32 Level::BytesPerSecondScale

6.54.1.2 card32 Level::FramesPerSecond

6.54.1.3 card32 Level::FramesPerSecondScale

6.54.1.4 const cardinal Level::MaxQualityLayers = 4 [static]

6.54.1.5 card32 Level::PacketsPerSecondScale

6.54.1.6 Layer Level::QualityLayer[MaxQualityLayers]

6.54.1.7 card8 Level::QualityLayers

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

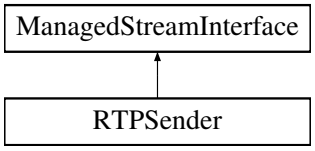
- advancedaudiopacket.cc

6.55 ManagedStreamInterface Class Reference

Managed Stream Interface.

`#include <managedstreaminterface.h>`

Inheritance diagram for ManagedStreamInterface:



Public Member Functions

- virtual ~ManagedStreamInterface ()
- virtual AbstractQoSDescription * getQoSDescription (const card64 offset)=0
- virtual void updateQuality (const AbstractQoSDescription *aqd)=0
- virtual void lock ()=0
- virtual void unlock ()=0

6.55.1 Detailed Description

Managed Stream Interface.

This is an interface for a bandwidth-managed stream.

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

6.55.2 Constructor & Destructor Documentation

6.55.2.1 virtual ManagedStreamInterface::~ManagedStreamInterface ( ) [inline, virtual]

Destructor.

6.55.3 Member Function Documentation

6.55.3.1 virtual AbstractQoSDescription* ManagedStreamInterface::getQoSDescription (const card64 *offset*) [pure virtual]

Get QoS description for current time + offset.

Parameters

offset	Offset in microseconds.

Returns

QoS Description.

Implemented in RTPSender.

6.55.3.2 virtual void ManagedStreamInterface::lock ( ) [pure virtual]

Lock stream.

Implemented in RTPSender.

6.55.3.3 virtual void ManagedStreamInterface::unlock ( ) [pure virtual]

Unlock stream.

Implemented in RTPSender.

6.55.3.4 virtual void ManagedStreamInterface::updateQuality (const AbstractQoSDescription * *aqd*) [pure virtual]

Update encoder's quality with changes made in QoS description returned b getQoS-Description().

Parameters

aqd	QoS Description.

See also

RTPSender::getQoSDescription

Implemented in RTPSender.

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

- managedstreaminterface.h

6.56 MediaInfo Struct Reference

Media Info.

`#include <mediainfo.h>`

Public Member Functions

- MediaInfo ()
- void reset ()
- void translate ()

Public Attributes

- card64 StartTimeStamp
- card64 EndTimeStamp
- char Title [MaxTitleLength+1]
- char Artist [MaxArtistLength+1]
- char Comment [MaxCommentLength+1]

Static Public Attributes

- static const cardinal MaxTitleLength = 47
- static const cardinal MaxArtistLength = 47
- static const cardinal MaxCommentLength = 47

6.56.1 Detailed Description

Media Info.

This class contains information on a media.

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

6.56.2 Constructor & Destructor Documentation

6.56.2.1 MediaInfo::MediaInfo ( )

Constructor.

6.56.3 Member Function Documentation

6.56.3.1 void MediaInfo::reset ( )

Reset.

6.56.3.2 void MediaInfo::translate ( )

Translate byte order.

6.56.4 Member Data Documentation

6.56.4.1 char MediaInfo::Artist[MaxArtistLength+1]

Artist string.

6.56.4.2 char MediaInfo::Comment[MaxCommentLength+1]

Comment string.

6.56.4.3 card64 MediaInfo::EndTimeStamp

End time stamp of the media.

6.56.4.4 const cardinal MediaInfo::MaxArtistLength = 47 [static]

Constant for the maximum author length.

6.56.4.5 const cardinal MediaInfo::MaxCommentLength = 47 [static]

Constant for the maximum comment length.

6.56.4.6 const cardinal MediaInfo::MaxTitleLength = 47 [static]

Constant for the maximum title length.

6.56.4.7 card64 MediaInfo::StartTimeStamp

Start time stamp of the media.

6.56.4.8 char MediaInfo::Title[MaxTitleLength+1]

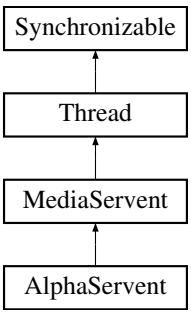
Title string.

The documentation for this struct was generated from the following files:

- mediainfo.h
- mediainfo.cc

6.57 MediaServent Class Reference

Inheritance diagram for MediaServent:



Public Member Functions

- MediaServent (AbstractMediaServer *server, const String &identifier, SocketAddress *peerAddress, const integer communicationDomain=Socket::IP,

const integer socketType=Socket::UDP, const integer socketProtocol=Socket::Default, const SocketAddress **localAddressArray=NULL, const cardinal localAddresses=0)
- ~MediaServent ()
- ServentQueueErrors queueRequest (AbstractMediaServentRequest *request)
- void updateReport (const MediaServentLayerReport &report, const cardinal layer)
- void shutdown (const ShutdownReason reason)
- const String getIdentifier () const
- void setTimeout (const card64 timeout)
- card64 getTimeout () const
- void updateTimeStamp (const card64 timeStamp=getMicroTime())
- bool hasTimedOut (const card64 now=getMicroTime()) const
- virtual bool transmissionErrorOccured ()=0

Protected Member Functions

- virtual void handleRequest (AbstractMediaServentRequest *request)=0
- void run ()

Protected Attributes

- AbstractMediaServer * Server
- String Identifier
- MessageQueue < AbstractMediaServentRequest > Queue
- card64 TimeStamp
- card64 Timeout
- MediaServentReport Report
- ShutdownReason ShutdownStatus
- Socket SenderSocket
- InternetFlow Flow
- card16 LastSequenceNumber
- card16 PosChgSeqNumber
- bool UserLimitPause
- bool ManagerLimitPause
- bool ClientPause

6.57.1 Constructor & Destructor Documentation

6.57.1.1 MediaServent::MediaServent (AbstractMediaServer * *server*, const String & *identifier*, SocketAddress * *peerAddress*, const integer *communicationDomain* = Socket::IP, const integer *socketType* = Socket::UDP, const integer *socketProtocol* = Socket::Default, const SocketAddress ** *localAddressArray* = NULL, const cardinal *localAddresses* = 0)

6.57.1.2 MediaServent::~MediaServent ( )

6.57.2 Member Function Documentation

6.57.2.1 const String MediaServent::getIdentifier ( ) const [inline]

6.57.2.2 card64 MediaServent::getTimeout ( ) const [inline]

6.57.2.3 virtual void MediaServent::handleRequest (AbstractMediaServentRequest * *request*) [protected, pure virtual]

Implemented in AlphaServent.

6.57.2.4 bool MediaServent::hasTimedOut (const card64 *now* = getMicroTime ()) const [inline]

6.57.2.5 ServentQueueErrors MediaServent::queueRequest (AbstractMediaServentRequest * *request*)

6.57.2.6 void MediaServent::run ( ) [protected, virtual]

The virtual run() method, which contains the thread's implementation. It has to be implemented by classes, which inherit Thread.

Implements Thread.

6.57.2.7 void MediaServent::setTimeout (const card64 *timeout*) [inline]

6.57.2.8 void MediaServent::shutdown (const ShutdownReason *reason*)

6.57.2.9 virtual bool MediaServent::transmissionErrorOccured ( ) [pure virtual]

Implemented in AlphaServent.

6.57.2.10 void MediaServent::updateReport (const MediaServentLayerReport & *report,* const cardinal *layer*)

6.57.2.11 void MediaServent::updateTimeStamp (const card64 *timeStamp* = getMicroTime ()) [inline]

6.57.3 Member Data Documentation

6.57.3.1 bool MediaServent::ClientPause [protected]

6.57.3.2 InternetFlow MediaServent::Flow [protected]

6.57.3.3 String MediaServent::Identifier [protected]

6.57.3.4 card16 MediaServent::LastSequenceNumber [protected]

6.57.3.5 bool MediaServent::ManagerLimitPause [protected]

6.57.3.6 card16 MediaServent::PosChgSeqNumber [protected]

6.57.3.7 MessageQueue<AbstractMediaServentRequest> MediaServent::Queue [protected]

6.57.3.8 MediaServentReport MediaServent::Report [protected]

6.57.3.9 Socket MediaServent::SenderSocket [protected]

6.57.3.10 AbstractMediaServer* MediaServent::Server [protected]

6.57.3.11 ShutdownReason MediaServent::ShutdownStatus [protected]

6.57.3.12 card64 MediaServent::Timeout [protected]

6.57.3.13 card64 MediaServent::TimeStamp [protected]

6.57.3.14 bool MediaServent::UserLimitPause [protected]

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

- s2.cc

6.58 MediaServentLayerReport Struct Reference

Public Attributes

- card64 LastUpdate
- double FractionLost
- double Jitter

6.58.1 Member Data Documentation

6.58.1.1 double MediaServentLayerReport::FractionLost

6.58.1.2 double MediaServentLayerReport::Jitter

6.58.1.3 card64 MediaServentLayerReport::LastUpdate

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

- s2.cc

6.59 MediaServentReport Struct Reference

Public Attributes

- cardinal Layers
- MediaServentLayerReport LayerReport [MaxMediaServentLayers]

6.59.1 Member Data Documentation

6.59.1.1 MediaServentLayerReport MediaServentReport::LayerReport[MaxMediaServentLayers]

6.59.1.2 cardinal MediaServentReport::Layers

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

- s2.cc

6.60 MessageQueue< T >::Message Struct Reference

Public Attributes

- Message * Next
- T * Content

template<class T> struct MessageQueue< T >::Message

6.60.1 Member Data Documentation

6.60.1.1 template<class T> T* MessageQueue< T >::Message::Content

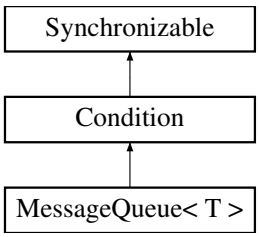
6.60.1.2 template<class T> Message* MessageQueue< T >::Message::Next

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

- s2.cc

6.61 MessageQueue< T > Class Template Reference

Inheritance diagram for MessageQueue< T >:



Classes

- struct Message

Public Member Functions

- MessageQueue ()
- ~MessageQueue ()
- void flush ()
- bool push (T *request)
- T * pop ()

Public Attributes

- Message * FirstMessage
- Message * LastMessage

template<class T> class MessageQueue< T >

6.61.1 Constructor & Destructor Documentation

6.61.1.1 template<class T > MessageQueue< T >::MessageQueue ( )

6.61.1.2 template<class T > MessageQueue< T >::~MessageQueue ( )

6.61.2 Member Function Documentation

6.61.2.1 template<class T > void MessageQueue< T >::flush ( )

6.61.2.2 template<class T > T * MessageQueue< T >::pop ( )

6.61.2.3 template<class T> bool MessageQueue< T >::push (T * *request*)

6.61.3 Member Data Documentation

6.61.3.1 template<class T> Message* MessageQueue< T >::FirstMessage

6.61.3.2 template<class T> Message* MessageQueue< T >::LastMessage

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

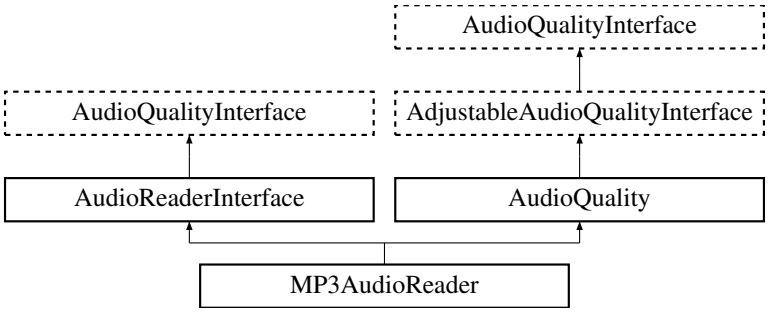
- s2.cc

6.62 MP3AudioReader Class Reference

MP3 Audio Reader.

`#include <mp3audioreader.h>`

Inheritance diagram for MP3AudioReader:



Public Member Functions

- MP3AudioReader (const char *name=NULL)
- ~MP3AudioReader ()
- bool openMedia (const char *name)
- void closeMedia ()
- bool ready () const
- void getMediaInfo (MediaInfo &mediaInfo) const
- MediaError getErrorCode () const
- card64 getPosition () const
- card64 getMaxPosition () const
- void setPosition (const card64 position)
- cardinal getNextBlock (void *buffer, const cardinal blockSize)

Private Member Functions

- bool initialize (char *filename)
- bool setsoundtype (int stereo, int samplesize, int speed)
- void set8bitmode ()

- bool putblock (void *buffer, int size)
- int putblock_nt (void *buffer, int size)
- void releasedevice ()
- bool attachdevice ()
- bool readNextFrame ()

Private Attributes

- Mpegtoraw * MP3Decoder
- Soundinputstreamfromfile * MP3Source
- cardinal BufferPos
- cardinal BufferSize
- double FramesPerSecond
- card64 Position
- card64 MaxPosition
- MediaError Error
- char Buffer [RAWDATASIZE *sizeof(short int)]

6.62.1 Detailed Description

MP3 Audio Reader.

This class is a reader for MP3 audio files.

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

6.62.2 Constructor & Destructor Documentation

6.62.2.1 MP3AudioReader::MP3AudioReader (const char * *name* = NULL)

Constructor.

Parameters

name	Name of MP3 file or NULL.

6.62.2.2 MP3AudioReader::~MP3AudioReader ( )

Destructor.

6.62.3 Member Function Documentation

6.62.3.1 bool MP3AudioReader::attachdevice ( ) [private]

6.62.3.2 void MP3AudioReader::closeMedia ( ) [virtual]

closeMedia() implementation of AudioReaderInterface.

See also

AudioReaderInterface::closeMedia

Implements AudioReaderInterface.

6.62.3.3 MediaError MP3AudioReader::getErrorCode ( ) const [virtual]

getErrorCode() implementation of AudioReaderInterface.

See also

AudioReaderInterface::getErrorCode

Implements AudioReaderInterface.

6.62.3.4 card64 MP3AudioReader::getMaxPosition ( ) const [virtual]

getMaxPosition() implementation of AudioReaderInterface.

See also

AudioReaderInterface::getMaxPosition

Implements AudioReaderInterface.

6.62.3.5 void MP3AudioReader::getMediaInfo (MediaInfo & *mediaInfo*) const [virtual]

getMediaInfo() implementation of AudioReaderInterface.

See also

AudioReaderInterface::getMediaInfo

Implements AudioReaderInterface.

6.62.3.6 cardinal MP3AudioReader::getNextBlock (void * *buffer,* const cardinal *blockSize*) [virtual]

getNextBlock() implementation of AudioReaderInterface.

See also

AudioReaderInterface::getNextBlock

Implements AudioReaderInterface.

6.62.3.7 card64 MP3AudioReader::getPosition ( ) const [virtual]

getPosition() implementation of AudioReaderInterface.

See also

AudioReaderInterface::getPosition

Implements AudioReaderInterface.

6.62.3.8 bool MP3AudioReader::initialize (char * *filename*) [private]

6.62.3.9 bool MP3AudioReader::openMedia (const char * *name*) [virtual]

openMedia() implementation of AudioReaderInterface.

See also

AudioReaderInterface::openMedia

Implements AudioReaderInterface.

6.62.3.10 bool MP3AudioReader::putblock (void * *buffer,* int *size*) [private]

6.62.3.11 int MP3AudioReader::putblock_nt (void * *buffer,* int *size*) [private]

6.62.3.12 bool MP3AudioReader::readNextFrame ( ) [private]

6.62.3.13 bool MP3AudioReader::ready ( ) const [virtual]

ready() implementation of AudioReaderInterface.

See also

AudioReaderInterface::ready

Implements AudioReaderInterface.

6.62.3.14 void MP3AudioReader::releasedevice ( ) [private]

6.62.3.15 void MP3AudioReader::set8bitmode ( ) [private]

6.62.3.16 void MP3AudioReader::setPosition (const card64 *position*) [virtual]

setPosition() implementation of AudioReaderInterface.

See also

AudioReaderInterface::setPosition

Implements AudioReaderInterface.

6.62.3.17 bool MP3AudioReader::setsoundtype (int *stereo,* int *samplesize,* int *speed*) [private]

6.62.4 Member Data Documentation

6.62.4.1 char MP3AudioReader::Buffer[RAWDATASIZE ∗sizeof(short int)] [private]

6.62.4.2 cardinal MP3AudioReader::BufferPos [private]

6.62.4.3 cardinal MP3AudioReader::BufferSize [private]

6.62.4.4 MediaError MP3AudioReader::Error [private]

6.62.4.5 double MP3AudioReader::FramesPerSecond [private]

6.62.4.6 card64 MP3AudioReader::MaxPosition [private]

6.62.4.7 Mpegtoraw∗ MP3AudioReader::MP3Decoder [private]

6.62.4.8 Soundinputstreamfromfile∗ MP3AudioReader::MP3Source [private]

6.62.4.9 card64 MP3AudioReader::Position [private]

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

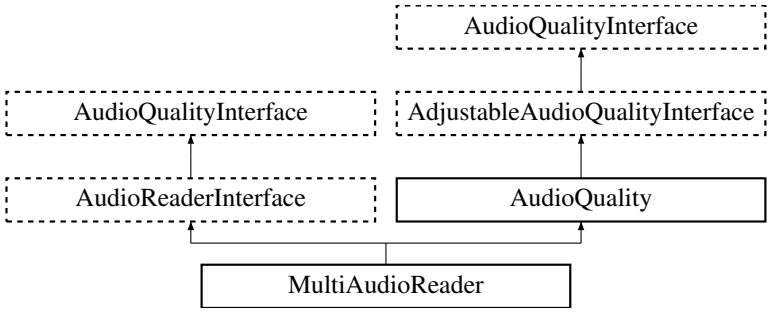
- mp3audioreader.h
- mp3audioreader.cc

6.63 MultiAudioReader Class Reference

Multi Audio Reader.

`#include <multiaudioreader.h>`

Inheritance diagram for MultiAudioReader:



Classes

- struct ReaderEntry

Public Member Functions

- MultiAudioReader (const char *name=NULL, const cardinal level=0)
- ~MultiAudioReader ()
- bool openMedia (const char *name)
- void closeMedia ()
- bool ready () const
- void getMediaInfo (MediaInfo &mediaInfo) const
- MediaError getErrorCode () const
- card64 getPosition () const
- card64 getMaxPosition () const
- void setPosition (card64 position)
- cardinal getNextBlock (void *buffer, const cardinal blockSize)
- AudioReaderInterface * getAudioReader (const char *name, const cardinal level)

Private Attributes

- AudioReaderInterface * Reader
- std::multimap< const card64, ReaderEntry > ReaderSet
- std::multimap< const card64, ReaderEntry >::iterator ReaderIterator
- MediaError Error
- card64 Position
- card64 MaxPosition
- cardinal Level

6.63.1 Detailed Description

Multi Audio Reader.

This class is a reader for multiple audio files from a list.

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

6.63.2 Constructor & Destructor Documentation

6.63.2.1 MultiAudioReader::MultiAudioReader (const char * *name* = NULL, const cardinal *level* = 0)

Constructor.

Parameters

name	Name of AudioList file or NULL.
level	Recursion level (normally 0).

6.63.2.2 MultiAudioReader::~MultiAudioReader ( )

Destructor.

6.63.3 Member Function Documentation

6.63.3.1 void MultiAudioReader::closeMedia ( ) [virtual]

closeMedia() implementation of AudioReaderInterface.

See also

AudioReaderInterface::closeMedia

Implements AudioReaderInterface.

6.63.3.2 AudioReaderInterface * MultiAudioReader::getAudioReader (const char * *name*, const cardinal *level*)

Get AudioReaderInterface for loading a given file.

Parameters

name	File name.
level	Recursion level (normally 0).

Returns

AudioReaderInterface, if load was successfull; NULL otherwise.

6.63.3.3 MediaError MultiAudioReader::getErrorCode ( ) const [virtual]

getErrorCode() implementation of AudioReaderInterface.

See also

AudioReaderInterface::getErrorCode

Implements AudioReaderInterface.

6.63.3.4 card64 MultiAudioReader::getMaxPosition ( ) const [virtual]

getMaxPosition() implementation of AudioReaderInterface.

See also

AudioReaderInterface::getMaxPosition

Implements AudioReaderInterface.

6.63.3.5 void MultiAudioReader::getMediaInfo (MediaInfo & *mediaInfo*) const [virtual]

getMediaInfo() implementation of AudioReaderInterface.

See also

AudioReaderInterface::getMediaInfo

Implements AudioReaderInterface.

6.63.3.6 cardinal MultiAudioReader::getNextBlock (void * *buffer*, const cardinal *blockSize*) [virtual]

getNextBlock() implementation of AudioReaderInterface.

See also

AudioReaderInterface::getNextBlock

Implements AudioReaderInterface.

6.63.3.7 card64 MultiAudioReader::getPosition ( ) const [virtual]

getPosition() implementation of AudioReaderInterface.

See also

AudioReaderInterface::getPosition

Implements AudioReaderInterface.

6.63.3.8 bool MultiAudioReader::openMedia (const char * *name*) [virtual]

openMedia() implementation of AudioReaderInterface.

See also

AudioReaderInterface::openMedia

Implements AudioReaderInterface.

6.63.3.9 bool MultiAudioReader::ready ( ) const [virtual]

ready() implementation of AudioReaderInterface.

See also

AudioReaderInterface::ready

Implements AudioReaderInterface.

6.63.3.10 void MultiAudioReader::setPosition (card64 *position*) [virtual]

setPosition() implementation of AudioReaderInterface.

See also

AudioReaderInterface::setPosition

Implements AudioReaderInterface.

6.63.4 Member Data Documentation

6.63.4.1 MediaError MultiAudioReader::Error [private]

6.63.4.2 cardinal MultiAudioReader::Level [private]

6.63.4.3 card64 MultiAudioReader::MaxPosition [private]

6.63.4.4 card64 MultiAudioReader::Position [private]

6.63.4.5 AudioReaderInterface* MultiAudioReader::Reader [private]

6.63.4.6 std::multimap<const card64, ReaderEntry>::iterator MultiAudioReader::ReaderIterator [private]

6.63.4.7 std::multimap<const card64, ReaderEntry> MultiAudioReader::ReaderSet [private]

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

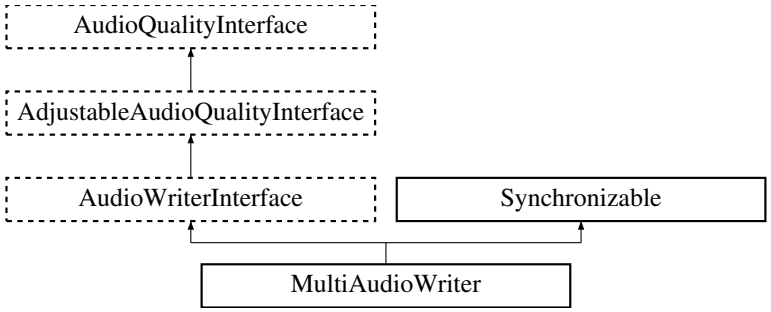
- multiaudioreader.h
- multiaudioreader.cc

6.64 MultiAudioWriter Class Reference

Multi Audio Writer.

`#include <multiaudiowriter.h>`

Inheritance diagram for MultiAudioWriter:



Public Member Functions

- MultiAudioWriter ()
- ~MultiAudioWriter ()
- bool addWriter (AudioWriterInterface *writer)
- void removeWriter (AudioWriterInterface *writer)
- card16 getSamplingRate () const
- card8 getBits () const
- card8 getChannels () const
- card16 getByteOrder () const
- card16 setSamplingRate (const card16 samplingRate)
- card8 setBits (const card8 bits)
- card8 setChannels (const card8 channels)

- card16 setByteOrder (const card16 byteOrder)
- cardinal getBytesPerSecond () const
- cardinal getBitsPerSample () const
- bool ready () const
- void sync ()
- bool write (const void *data, const size_t length)

Private Attributes

- std::multiset < AudioWriterInterface * > WriterSet
- card16 AudioSamplingRate
- card8 AudioBits
- card8 AudioChannels
- card16 AudioByteOrder

6.64.1 Detailed Description

Multi Audio Writer.

This class implements AudioWriterInterface for a set of AudioWriterInterfaces. Example: AudioDevice + AudioDebug + SpectrumAnalyzer.

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

6.64.2 Constructor & Destructor Documentation

6.64.2.1 MultiAudioWriter::MultiAudioWriter ( )

Constructor.

6.64.2.2 MultiAudioWriter::~MultiAudioWriter ( )

Destructor.

6.64.3 Member Function Documentation

6.64.3.1 bool MultiAudioWriter::addWriter (AudioWriterInterface * *writer*)

Add new AudioWriterInferface to writer set.

Parameters

writer	AudioWriterInterface object.

Returns

true, if writer has been added; false otherwise.

6.64.3.2 card8 MultiAudioWriter::getBits ( ) const [virtual]

getBits() Implementation of AudioQualityInterface.

See also

AudioQualityInterface::getBits

Implements AudioQualityInterface.

6.64.3.3 cardinal MultiAudioWriter::getBitsPerSample ( ) const [virtual]

getBitsPerSample() Implementation of AudioQualityInterface.

See also

AudioQualityInterface::getBitsPerSample

Implements AudioQualityInterface.

6.64.3.4 card16 MultiAudioWriter::getByteOrder ( ) const [virtual]

getByteOrder() Implementation of AudioQualityInterface.

See also

AudioQualityInterface::getByteOrder

Implements AudioQualityInterface.

6.64.3.5 cardinal MultiAudioWriter::getBytesPerSecond ( ) const [virtual]

getBytesPerSecond() Implementation of AudioQualityInterface.

See also

AudioQualityInterface::getBytesPerSecond

Implements AudioQualityInterface.

6.64.3.6 card8 MultiAudioWriter::getChannels ( ) const [virtual]

getChannels() Implementation of AudioQualityInterface.

See also

AudioQualityInterface::getChannels

Implements AudioQualityInterface.

6.64.3.7 card16 MultiAudioWriter::getSamplingRate ( ) const [virtual]

getSamplingRate() Implementation of AudioQualityInterface.

See also

AudioQualityInterface::getSamplingRate

Implements AudioQualityInterface.

6.64.3.8 bool MultiAudioWriter::ready ( ) const [virtual]

ready() implementation of AudioWriterInterface

See also

AudioWriterInterface::ready

Implements AudioWriterInterface.

6.64.3.9 void MultiAudioWriter::removeWriter (AudioWriterInterface * *writer*)

Remove AudioWriterInterface object from writer set.

6.64.3.10 card8 MultiAudioWriter::setBits (const card8 *bits*) [virtual]

setBits() Implementation of AudioQualityInterface.

See also

AudioQualityInterface::setBits

Implements AdjustableAudioQualityInterface.

6.64.3.11 card16 MultiAudioWriter::setByteOrder (const card16 *byteOrder*) [virtual]

setByteOrder() Implementation of AudioQualityInterface.

See also

AudioQualityInterface::setByteOrder

Implements AdjustableAudioQualityInterface.

6.64.3.12 card8 MultiAudioWriter::setChannels (const card8 *channels*) [virtual]

setChannels() Implementation of AudioQualityInterface.

See also

AudioQualityInterface::setChannels

Implements AdjustableAudioQualityInterface.

6.64.3.13 card16 MultiAudioWriter::setSamplingRate (const card16 *samplingRate*) [virtual]

setSamplingRate() Implementation of AudioQualityInterface.

See also

AudioQualityInterface::setSamplingRate

Implements AdjustableAudioQualityInterface.

6.64.3.14 void MultiAudioWriter::sync ( ) [virtual]

sync() implementation of AudioWriterInterface

See also

AudioWriterInterface::sync

Implements AudioWriterInterface.

6.64.3.15 bool MultiAudioWriter::write (const void * *data*, const size_t *length*) [virtual]

write() implementation of AudioWriterInterface

See also

AudioWriterInterface::write

Implements AudioWriterInterface.

6.64.4 Member Data Documentation

6.64.4.1 card8 MultiAudioWriter::AudioBits [private]

6.64.4.2 card16 MultiAudioWriter::AudioByteOrder [private]

6.64.4.3 card8 MultiAudioWriter::AudioChannels [private]

6.64.4.4 card16 MultiAudioWriter::AudioSamplingRate [private]

6.64.4.5 std::multiset<AudioWriterInterface*> MultiAudioWriter::WriterSet [private]

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

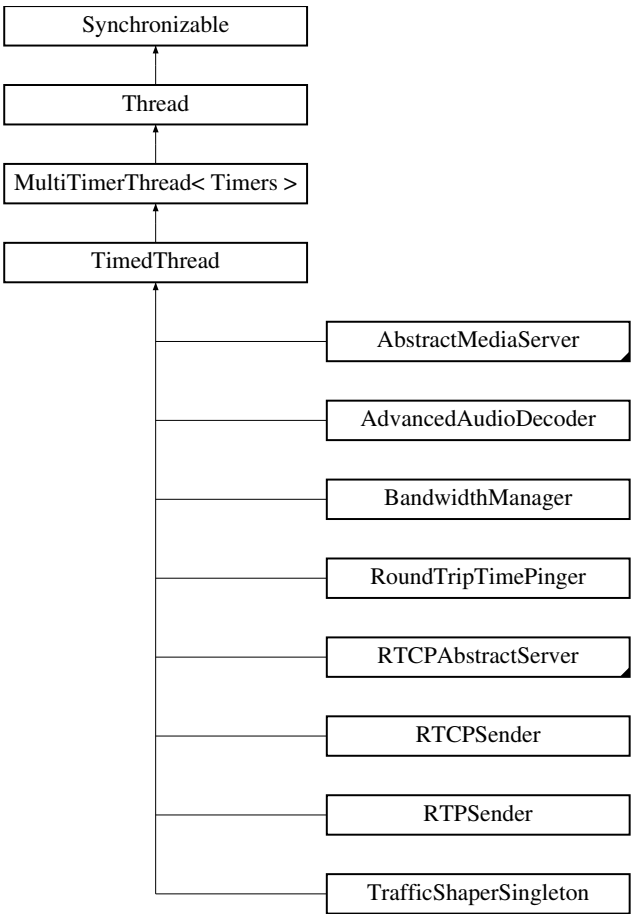
- multiaudiowriter.h
- multiaudiowriter.cc

6.65 MultiTimerThread< Timers > Class Template Reference

Multi Timer Thread.

`#include <multitimerthread.h>`

Inheritance diagram for MultiTimerThread< Timers >:



Classes

- struct TimerParameters

Public Member Functions

- MultiTimerThread (const char *name="MultiTimerThread", const cardinal flags=-TF_CancelDeferred)
- ~MultiTimerThread ()
- card64 getInterval (const cardinal timer)
- void setInterval (const cardinal timer, const card64 usec, const card64 callLimit=0)
- void setNextAction (const cardinal timer, const card64 usec=0, const card64 callLimit=1)
- void setNextActionAbs (const cardinal timer, const card64 timeStamp=0, const card64 callLimit=1)
- cardinal getTimerCorrection (const cardinal timer)

- void setTimerCorrection (const cardinal timer, const cardinal maxCorrection=0)
- void leaveCorrectionLoop (const cardinal timer)
- void setFastStart (const cardinal timer, const bool on)
- bool getFastStart (const cardinal timer) const
- void cancel ()
- void ∗ stop ()

Protected Member Functions

- virtual void timerEvent (const cardinal timer)=0

Private Member Functions

- void run ()
- bool isShuttingDown ()

Private Attributes

- TimerParameters Parameters [Timers]
- bool ParametersUpdated
- bool Shutdown
- bool LeaveCorrectionLoop [Timers]

Static Private Attributes

- static const card64 UpdateResolution = 100000

6.65.1 Detailed Description

template<const cardinal Timers>class MultiTimerThread< Timers >

Multi Timer Thread.

This abstract class realizes a timer thread with multiple timers, based on Thread. The user of this class has to implement timerEvent(). Inaccurate system timers are corrected by calling user's timerEvent() implementation multiple times if necessary. This feature can be modified by setTimerCorrection (Default is on at a maximum of 10 calls).

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

See also

Thread

6.65.2 Constructor & Destructor Documentation

6.65.2.1 template<const cardinal Timers> MultiTimerThread< Timers >::MultiTimerThread (const char * *name* = "MultiTimerThread< Timers >", const cardinal *flags* = TF_CancelDeferred)

Constructor. A new multitimer thread with a given interval will be created but *not* started! To start the new thread, call start(). The interval gives the time for the interval in microseconds, the virtual function timerEvent() is called. The default timer correction is set to 10. See setTimerCorrection() for more information on timer correction. The first call of timerEvent() will be made immediately, if the fast start option is set (default). Otherwise it will be made after the given interval.

Parameters

usec	Interval in microseconds.
name	Thread name.
flags	Thread flags.

See also

Thread::start
timerEvent
Thread::Thread
setTimerCorrection
setFastStart

6.65.2.2 template<const cardinal Timers> MultiTimerThread< Timers >::~MultiTimerThread ( )

Destructor.

6.65.3 Member Function Documentation

6.65.3.1 template<const cardinal Timers> void MultiTimerThread< Timers >::cancel ( ) [virtual]

Reimplementation of Thread's cancel() method.

See also

Thread::cancel

Reimplemented from Thread.

6.65.3.2 template<const cardinal Timers> bool MultiTimerThread< Timers >::getFastStart (const cardinal *timer*) const [inline]

Get fast start option: If false, the first call of timerEvent() will be made ∗after∗ the given interval; otherwise it will be made immediately.

Parameters

timer	Timer number.

Returns

true, if option is set; false otherwise.

6.65.3.3 template<const cardinal Timers> card64 MultiTimerThread< Timers >::getInterval (const cardinal *timer*) [inline]

Get timer interval.

Parameters

timer	Timer number.

Returns

Interval in microseconds.

6.65.3.4 template<const cardinal Timers> cardinal MultiTimerThread< Timers >::getTimerCorrection (const cardinal *timer*) [inline]

Get maximum correction value for inaccurate system timer.

Parameters

timer	Timer number.

Returns

true, if activated; false if not.

See also

setTimerCorrection

6.65.3.5 template<const cardinal Timers> bool MultiTimerThread< Timers >::isShuttingDown ( ) [inline, private]

6.65.3.6 template<const cardinal Timers> void MultiTimerThread< Timers >::leaveCorrectionLoop (const cardinal *timer*) [inline]

Leave timer correction loop: If the thread is in a timer correction loop, the loop will be finished after the current timerEvent() call returns.

Parameters

timer	Timer number.

6.65.3.7 template<const cardinal Timers> void MultiTimerThread< Timers >::run ( ) [private, virtual]

The virtual run() method, which contains the thread's implementation. It has to be implemented by classes, which inherit Thread.

Implements Thread.

6.65.3.8 template<const cardinal Timers> void MultiTimerThread< Timers >::setFastStart (const cardinal *timer,* const bool *on*) [inline]

Set fast start option: If false, the first call of timerEvent() will be made ∗after∗ the given interval; otherwise it will be made immediately. The default is true.

Parameters

timer	Timer number.
on	true, to set option; false otherwise.

6.65.3.9 template<const cardinal Timers> void MultiTimerThread< Timers >::setInterval (const cardinal *timer,* const card64 *usec,* const card64 *callLimit* = 0) [inline]

Set timer interval. Note, that the first timerEvent() call will be immediately, is FastStart mode is set, see also setFastStart(). For single shot timers, you probably have to call setFastStart(nr,0) first!

Parameters

timer	Timer number.
usec	Interval in microseconds (0 to deactivate timer).
callLimit	Call count limit (0 for infinite).

See also

setFastStart

6.65.3.10 template<const cardinal Timers> void MultiTimerThread< Timers >::setNextAction (const cardinal *timer*, const card64 *usec* = 0, const card64 *callLimit* = 1) [inline]

Like setInterval(), but disabling FastStart first. This method can be used e.g. for single shot timers.

Parameters

timer	Timer number.
usec	Time to next invokation (0 = immediately).
callLimit	Call count limit (0 for infinite, default: 1).

See also

setInterval

6.65.3.11 template<const cardinal Timers> void MultiTimerThread< Timers >::setNextActionAbs (const cardinal *timer*, const card64 *timeStamp* = 0, const card64 *callLimit* = 1) [inline]

Like setNextAction(), but the time stamp of the next invokation is given as absolute time (microseconds since January 01, 1970).

Parameters

timer	Timer number.
usec	Time to next invokation (0 = immediately).
callLimit	Call count limit (0 for infinite, default: 1).

See also

setInterval
setNextAction

6.65.3.12 template<const cardinal Timers> void MultiTimerThread< Timers >::setTimerCorrection (const cardinal *timer*, const cardinal *maxCorrection* = 0) [inline]

Set correction of inaccurate system timer to given value. This on will cause the timerEvent() function to be called a maximum of maxCorrection times, if the total number of calls is lower than the calculated number of times the function should have been called. If the number of correction calls is higher than maxCorrection, ∗no∗ correction will be done! Default is 0, which turns correction off.

Parameters

timer	Timer number.
of	true to activate correction; false to deactivate.

6.65.3.13 template<const cardinal Timers> void* MultiTimerThread< Timers >::stop ( ) [virtual]

Reimplementation of Thread's stop() method.

See also

Thread::stop

Reimplemented from Thread.

Reimplemented in AbstractMediaServer, and RTCPAbstractServer.

6.65.3.14 template<const cardinal Timers> virtual void MultiTimerThread< Timers >::timerEvent (const cardinal *timer*) [protected, pure virtual]

The virtual timerEvent() method, which contains the multitimer thread's implementation. It has to be implemented by classes, which inherit MultiTimerThread. This method is called regularly with the given interval.

Implemented in TimedThread.

6.65.4 Member Data Documentation

6.65.4.1 template<const cardinal Timers> bool MultiTimerThread< Timers >::LeaveCorrectionLoop[Timers] [private]

6.65.4.2 template<const cardinal Timers> TimerParameters MultiTimerThread< Timers >::Parameters[Timers] [private]

6.65.4.3 template<const cardinal Timers> bool MultiTimerThread< Timers >::ParametersUpdated [private]

6.65.4.4 template<const cardinal Timers> bool MultiTimerThread< Timers >::Shutdown [private]

6.65.4.5 template<const cardinal Timers> const card64 MultiTimerThread< Timers >::UpdateResolution = 100000 [static, private]

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

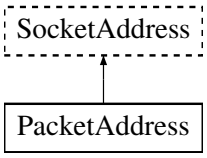
- multitimerthread.h

6.66 PacketAddress Class Reference

Packet Address.

`#include <packetaddress.h>`

Inheritance diagram for PacketAddress:



Public Member Functions

- PacketAddress ()
- PacketAddress (const PacketAddress &address)
- PacketAddress (const String &name)
- PacketAddress (const sockaddr *address, cardinal length)
- ~PacketAddress ()
- void reset ()
- SocketAddress * duplicate () const
- void init (const PacketAddress &address)
- void init (const String &name)
- PacketAddress & operator= (const PacketAddress &source)
- bool isValid () const
- integer getFamily () const
- String getAddressString (const cardinal format=PF_Default) const
- bool isNull () const
- card16 getPort () const
- void setPort (const card16 port)
- cardinal getSystemAddress (sockaddr *buffer, const socklen_t length, const cardinal type) const
- bool setSystemAddress (const sockaddr *address, const socklen_t length)
- int operator== (const PacketAddress &address) const
- int operator!= (const PacketAddress &address) const
- int operator< (const PacketAddress &address) const
- int operator<= (const PacketAddress &address) const
- int operator> (const PacketAddress &address) const
- int operator>= (const PacketAddress &address) const

Private Attributes

- char Name [MaxNameLength+1]

Static Private Attributes

- static const cardinal MaxNameLength = IFNAMSIZ - 1

6.66.1 Detailed Description

Packet Address.

This class manages a packet socket address.

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

6.66.2 Constructor & Destructor Documentation

6.66.2.1 PacketAddress::PacketAddress ( )

Constructor for an empty packet address.

6.66.2.2 PacketAddress::PacketAddress (const PacketAddress & *address*)

Constructor for an packet address from an packet address.

Parameters

address	Packet address.

6.66.2.3 PacketAddress::PacketAddress (const String & *name*)

Constructor for a packet address given by a string. Examples: "/tmp/test.socket".

Parameters

name	Address string.

6.66.2.4 PacketAddress::PacketAddress (const sockaddr * *address*, cardinal *length*)

Constructor for a packet address from the system's sockaddr structure.

Parameters

address	sockaddr.
length	Length of sockaddr.

6.66.2.5 PacketAddress::~PacketAddress ( )

Destructor.

6.66.3 Member Function Documentation

6.66.3.1 SocketAddress * PacketAddress::duplicate ( ) const [virtual]

duplicate() implementation of SocketAddress.

See also

SocketAddress::duplicate

Implements SocketAddress.

6.66.3.2 String PacketAddress::getAddressString (const cardinal *format* = PF_Default) const [virtual]

getAddressString() implementation of SocketAddress.

See also

SocketAddress::getAddress

Implements SocketAddress.

6.66.3.3 integer PacketAddress::getFamily ( ) const [virtual]

getFamily() implementation of SocketAddress.

See also

SocketAddress::getFamily

Implements SocketAddress.

6.66.3.4 card16 PacketAddress::getPort ( ) const [virtual]

getPort() implementation of SocketAddress.

See also

SocketAddress::getPort

Implements SocketAddress.

6.66.3.5 cardinal PacketAddress::getSystemAddress (sockaddr * *buffer,* const socklen_t *length,* const cardinal *type*) const [virtual]

getSystemAddress() implementation of SocketAddress

See also

SocketAddress::getSystemAddress

Implements SocketAddress.

6.66.3.6 void PacketAddress::init (const PacketAddress & *address*)

Initialize packet address from packet address.

6.66.3.7 void PacketAddress::init (const String & *name*)

Initialize packet address from socket name.

6.66.3.8 bool PacketAddress::isNull ( ) const [inline]

Check, if the address is null.

Returns

true, if the address is not null; false otherwise.

6.66.3.9 bool PacketAddress::isValid ( ) const [virtual]

isValid() implementation of SocketAddress.

See also

SocketAddress::isValid

Implements SocketAddress.

6.66.3.10 int PacketAddress::operator!= (const PacketAddress & *address*) const [inline]

Implementation of != operator.

6.66.3.11 int PacketAddress::operator< (const PacketAddress & *address*) const

Implementation of < operator.

6.66.3.12 int PacketAddress::operator<= (const PacketAddress & *address*) const [inline]

Implementation of <= operator.

6.66.3.13 PacketAddress& PacketAddress::operator= (const PacketAddress & *source*) [inline]

Implementation of = operator.

6.66.3.14 int PacketAddress::operator== (const PacketAddress & *address*) const

Implementation of == operator.

6.66.3.15 int PacketAddress::operator> (const PacketAddress & *address*) const

Implementation of > operator.

6.66.3.16 int PacketAddress::operator>= (const PacketAddress & *address*) const [inline]

Implementation of >= operator.

6.66.3.17 void PacketAddress::reset ( ) [virtual]

Reset packet address.

Implements SocketAddress.

6.66.3.18 void PacketAddress::setPort (const card16 *port*) [virtual]

setPort() implementation of SocketAddress.

See also

SocketAddress::setPort

Implements SocketAddress.

6.66.3.19 bool PacketAddress::setSystemAddress (const sockaddr * *address*, const socklen_t *length*) [virtual]

setSystemAddress() implementation of SocketAddress.

See also

SocketAddress::setSystemAddress

Implements SocketAddress.

6.66.4 Member Data Documentation

6.66.4.1 const cardinal PacketAddress::MaxNameLength = IFNAMSIZ - 1 [static, private]

6.66.4.2 char PacketAddress::Name[MaxNameLength+1] [private]

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

- packetaddress.h
- packetaddress.cc

6.67 RoundTripTimePinger::Ping4Packet Struct Reference

Public Attributes

- icmp Header
- card64 TimeStamp

6.67.1 Member Data Documentation

6.67.1.1 icmp RoundTripTimePinger::Ping4Packet::Header

6.67.1.2 card64 RoundTripTimePinger::Ping4Packet::TimeStamp

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

- roundtriptimepinger.h

6.68 RoundTripTimePinger::Ping6Packet Struct Reference

Public Attributes

- icmp6_hdr Header

- card64 TimeStamp

6.68.1 Member Data Documentation

6.68.1.1 icmp6_hdr RoundTripTimePinger::Ping6Packet::Header

6.68.1.2 card64 RoundTripTimePinger::Ping6Packet::TimeStamp

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

- roundtriptimepinger.h

6.69 PingerHost Struct Reference

PingerHost.

```
#include <pingerhost.h>
```

Public Attributes

- InternetAddress Address
- String AddressString
- card64 LastPingTimeStamp
- card64 LastEchoTimeStamp
- cardinal RoundTripTime
- cardinal MaxRawRoundTripTime
- cardinal UserCount
- card16 SeqNum
- card8 TrafficClass
- bool IsIPv6

6.69.1 Detailed Description

PingerHost.

This structure contains internal information for RoundTripTimePinger.

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

6.69.2 Member Data Documentation

6.69.2.1 InternetAddress PingerHost::Address

InternetAddress to send ping to.

6.69.2.2 String PingerHost::AddressString

The ping address in text format.

6.69.2.3 bool PingerHost::IsIPv6

Does this address use IPv6?

6.69.2.4 card64 PingerHost::LastEchoTimeStamp

Timestamp of last received ping.

6.69.2.5 card64 PingerHost::LastPingTimeStamp

Timestamp of last sent ping.

6.69.2.6 cardinal PingerHost::MaxRawRoundTripTime

Maximum raw round trip time (directly calculated from packet).

6.69.2.7 cardinal PingerHost::RoundTripTime

Round trip time.

6.69.2.8 card16 PingerHost::SeqNum

Sequence number.

6.69.2.9 card8 PingerHost::TrafficClass

Traffic class.

6.69.2.10 cardinal PingerHost::UserCount

User counter (number of addHost() calls for this destination).

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

- pingerhost.h

6.70 PortableAddress Class Reference

Portable Internet Address.

`#include <portableaddress.h>`

Public Member Functions

- int operator== (const PortableAddress &address) const
- int operator!= (const PortableAddress &address) const
- int operator< (const PortableAddress &address) const
- int operator<= (const PortableAddress &address) const
- int operator> (const PortableAddress &address) const
- int operator>= (const PortableAddress &address) const
- void reset ()

Public Attributes

- card16 Host [8]
- card16 Port

6.70.1 Detailed Description

Portable Internet Address.

Binary representation for a socket address for sending the address over a network. The difference between InternetAddress is that PortableAddress does not contain hidden information on virtual function management, which make network transfer of InternetAddress objects problematic.

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

6.70.2 Member Function Documentation

6.70.2.1 int PortableAddress::operator!= (const PortableAddress & *address*) const

Implementation of == operator.

6.70.2.2 int PortableAddress::operator< (const PortableAddress & *address*) const

Implementation of < operator.

6.70.2.3 int PortableAddress::operator<= (const PortableAddress & *address*) const

Implementation of <= operator.

6.70.2.4 int PortableAddress::operator== (const PortableAddress & *address*) const

Implementation of == operator.

6.70.2.5 int PortableAddress::operator> (const PortableAddress & *address*) const

Implementation of > operator.

6.70.2.6 int PortableAddress::operator>= (const PortableAddress & *address*) const

Implementation of >= operator.

6.70.2.7 void PortableAddress::reset ( ) [inline]

Reset portable address.

6.70.3 Member Data Documentation

6.70.3.1 card16 PortableAddress::Host[8]

Host address in network byte order. IPv4 addresses are converted to IPv4-mapped IPv6 addresses.

6.70.3.2 card16 PortableAddress::Port

Port number.

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

- portableaddress.h

6.71 QAudioMixer Class Reference

QAudioMixer.

```
#include <qaudiomixer.h>
```

Public Slots

- void balance (int value)
- void volume (int value)
- void updateVolumeFromDevice ()
- void centerBalance ()
- void mute ()

Signals

- void closeAudioMixer ()

Public Member Functions

- QAudioMixer (AudioMixer ∗mixer, QWidget ∗parent=NULL)
- ∼QAudioMixer ()

Private Member Functions

- void closeEvent (QCloseEvent ∗event)
- void setVolumeOnDevice ()
- void updateText (const card8 left, const card8 right)

Private Attributes

- integer VolumeSetting
- integer BalanceSetting
- AudioMixer ∗ Mixer
- QPushButton ∗ Mute
- QSlider ∗ Balance
- QSlider ∗ Volume
- QLabel ∗ Values

6.71.1 Detailed Description

QAudioMixer.

This class is a Qt GUI for the audio mixer.

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

6.71.2 Constructor & Destructor Documentation

6.71.2.1 QAudioMixer::QAudioMixer (AudioMixer * *mixer*, QWidget * *parent* = NULL)

Constructor.

Parameters

mixer	AudioMixer object.
parent	Parent widget.

6.71.2.2 QAudioMixer::~QAudioMixer ( )

Destructor.

6.71.3 Member Function Documentation

6.71.3.1 void QAudioMixer::balance (int *value*) [slot]

Qt slot: Change balance.

6.71.3.2 void QAudioMixer::centerBalance ( ) [slot]

Qt slot: Center balance slider.

6.71.3.3 void QAudioMixer::closeAudioMixer ( ) [signal]

Qt signal: Emitted, when "Close" or window's close button is clicked.

6.71.3.4 void QAudioMixer::closeEvent (QCloseEvent * *event*) [private]

6.71.3.5 void QAudioMixer::mute ( ) [slot]

Qt slot: Mute.

6.71.3.6 void QAudioMixer::setVolumeOnDevice ( ) [private]

6.71.3.7 void QAudioMixer::updateText (const card8 *left*, const card8 *right*) [private]

6.71.3.8 void QAudioMixer::updateVolumeFromDevice ( ) [slot]

Qt slot: Update volume.

6.71.3.9 void QAudioMixer::volume (int *value*) [slot]

Qt slot: Change volume.

6.71.4 Member Data Documentation

6.71.4.1 QSlider* QAudioMixer::Balance [private]

6.71.4.2 integer QAudioMixer::BalanceSetting [private]

6.71.4.3 AudioMixer* QAudioMixer::Mixer [private]

6.71.4.4 QPushButton* QAudioMixer::Mute [private]

6.71.4.5 QLabel* QAudioMixer::Values [private]

6.71.4.6 QSlider* QAudioMixer::Volume [private]

6.71.4.7 integer QAudioMixer::VolumeSetting [private]

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

- qaudiomixer.h
- qaudiomixer.cc
- qaudiomixer_moc.cc

6.72 QClient Class Reference

QClient.

`#include <rtpa-qclient.h>`

Public Slots

- void play ()
- void stop ()
- void information ()
- void whatsThis ()
- void pause (bool on)
- void togglePause ()
- void toggleAddressResolution (bool selected)
- void spectrumAnalyzer ()
- void audioMixer ()
- void closeSpectrumAnalyzer ()
- void closeAudioMixer ()

- void quit ()
- void position (int value)
- void setSamplingRate (int index)
- void setChannels (bool stereo)
- void setBits (int index)
- void setEncoding (int index)
- void locationSelected (QAction *action)
- void loadBookmarks ()
- void clearBookmarks ()
- void saveBookmarks ()
- void timerEvent ()

Public Member Functions

- QClient (AudioWriterInterface *audioOutput, const char *receiverName=NULL, const char *defaultURL=NULL, SpectrumAnalyzer *analyzer=NULL, AudioMixer *mixer=NULL, QWidget *parent=NULL)
- ~QClient ()

Private Member Functions

- void updateCounter (card64 position)
- void insertURL (const String &urlToInsert, const int where=0)
- void showError (const cardinal error)
- QString bytesToQString (const card64 bytes) const
- QString card64ToQString (const card64 value, const char *formatString="%Ld") const
- QString doubleToQString (const double value, const char *formatString="%f") const
- QString flowInfoToQString (const card8 trafficClass, const card32 flowLabel) const

Private Attributes

- QAction * ResolverAction
- QAction * MixerAction
- QAction * SpectrumAnalyzerAction
- QAction * AutoRepeatAction
- QAction * AutoSaveBookmarksAction
- QAction * LocationAction [LocationCount]
- QSpectrumAnalyzer * SpectrumAnalyzerWindow
- QAudioMixer * MixerWindow
- SpectrumAnalyzer * SpectrumAnalyzerDevice
- AudioMixer * MixerDevice
- QMenu * ToolsMenu

- QMenu * URLMenu
- QMenu * SettingsMenu
- QLineEdit * Location
- QLCDNumber * Counter
- QLabel * TitleLabel
- QLabel * ArtistLabel
- QLabel * CommentLabel
- QLabel * StatusBar
- QInfoTabWidget * InfoWidget
- QInfoWidget * LayerInfo [MaxLayerInfo]
- QScrollBar * ScrollBar
- bool ScrollBarUpdated
- cardinal ScrollBarUpdateDelay
- cardinal EOFRepeatDelay
- QPushButton * Pause
- QWhatsThis * WhatsThis
- AudioClient * Client
- QList< String * > URLList
- String PlayingURL
- bool InsertionRequired

Static Private Attributes

- static const cardinal LocationCount = 15
- static const cardinal DisplayUpdateInterval = 250
- static const cardinal EOFRepeatInterval = 20000 / DisplayUpdateInterval
- static const cardinal MaxScrollBarUpdateDelay = 4
- static const cardinal MaxLayerInfo = 3

6.72.1 Detailed Description

QClient.

This class is the Qt-Toolkit GUI for the RTP audio client.

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

6.72.2 Constructor & Destructor Documentation

6.72.2.1 QClient::QClient (AudioWriterInterface * *audioOutput*, const char * *receiverName* = NULL, const char * *defaultURL* = NULL, SpectrumAnalyzer * *analyzer* = NULL, AudioMixer * *mixer* = NULL, QWidget * *parent* = NULL)

Constructor for new QClient.

Parameters

audioOutput	AudioWriter.
receiver-Name	Receiver name (e.g. ipv6-gaffel:1234); default NULL.
defaultUTL	Default URL (e.g. rtpa://gaffel:7500/Test.list); default NULL.
analyzer	SpectrumAnalyzer object; default NULL.
mixer	AudioMixer object; default NULL.
parent	Parent QWidget; default NULL.

6.72.2.2 QClient::~QClient ( )

Destructor.

6.72.3 Member Function Documentation

6.72.3.1 void QClient::audioMixer ( ) [slot]

Slot for "Audio Mixer" menu item.

6.72.3.2 QString QClient::bytesToQString (const card64 *bytes*) const [private]

6.72.3.3 QString QClient::card64ToQString (const card64 *value*, const char * *formatString* = "%Ld") const [private]

6.72.3.4 void QClient::clearBookmarks ( ) [slot]

Slot for removing all bookmarks.

6.72.3.5 void QClient::closeAudioMixer ( ) [slot]

Slot for close button of QAudioMixer.

6.72.3.6 void QClient::closeSpectrumAnalyzer ( ) [slot]

Slot for close button of QSpectrumAnalyzer.

6.72.3.7 QString QClient::doubleToQString (const double *value*, const char * *formatString* = "%f") const [private]

6.72.3.8 QString QClient::flowInfoToQString (const card8 *trafficClass*, const card32 *flowLabel*) const [private]

6.72.3.9 void QClient::information ( ) [slot]

Slot for "Information" button.

6.72.3.10 void QClient::insertURL (const String & *urlToInsert*, const int *where* = 0) [private]

6.72.3.11 void QClient::loadBookmarks ( ) [slot]

Slot for loading bookmarks.

6.72.3.12 void QClient::locationSelected (QAction * *action*) [slot]

Slot for location menu item.

6.72.3.13 void QClient::pause (bool *on*) [slot]

Slot for "Pause" button.

6.72.3.14 void QClient::play ( ) [slot]

Slot for "Play" button.

6.72.3.15 void QClient::position (int *value*) [slot]

Slot for position scrollbar.

6.72.3.16 void QClient::quit ( ) [slot]

Slot for "Quit" menu item.

6.72.3.17 void QClient::saveBookmarks ( ) [slot]

Slot for saving bookmarks.

6.72.3.18 void QClient::setBits (int *index*) [slot]

Slot for "Bits" combobox.

6.72.3.19 void QClient::setChannels (bool *stereo*) [slot]

Slot for "Stereo" checkbox.

6.72.3.20 void QClient::setEncoding (int *index*) [slot]

Slot for "Encoding" combobox.

6.72.3.21 void QClient::setSamplingRate (int *index*) [slot]

Slot for sampling rate combobox.

6.72.3.22 void QClient::showError (const cardinal *error*) [private]

6.72.3.23 void QClient::spectrumAnalyzer ( ) [slot]

Slot for "Spectrum Analyzer" menu item.

6.72.3.24 void QClient::stop ( ) [slot]

Slot for "Stop" button.

6.72.3.25 void QClient::timerEvent ( ) [slot]

Slot for QTimer.

6.72.3.26 void QClient::toggleAddressResolution (bool *selected*) [slot]

Slot for "Resolve Addresses" menu item.

6.72.3.27 void QClient::togglePause ( ) [slot]

Slot for "Toggle Pause" menu item.

6.72.3.28 void QClient::updateCounter (card64 *position*) [private]

6.72.3.29 void QClient::whatsThis ( ) [slot]

Slot for What's This mode.

6.72.4 Member Data Documentation

6.72.4.1 QLabel* QClient::ArtistLabel [private]

6.72.4.2 QAction* QClient::AutoRepeatAction [private]

6.72.4.3 QAction* QClient::AutoSaveBookmarksAction [private]

6.72.4.4 AudioClient* QClient::Client [private]

6.72.4.5 QLabel* QClient::CommentLabel [private]

6.72.4.6 QLCDNumber* QClient::Counter [private]

6.72.4.7 const cardinal QClient::DisplayUpdateInterval = 250 [static, private]

6.72.4.8 cardinal QClient::EOFRepeatDelay [private]

6.72.4.9 const cardinal QClient::EOFRepeatInterval = 20000 / DisplayUpdateInterval [static, private]

6.72.4.10 QInfoTabWidget* QClient::InfoWidget [private]

6.72.4.11 bool QClient::InsertionRequired [private]

6.72.4.12 QInfoWidget* QClient::LayerInfo[MaxLayerInfo] [private]

6.72.4.13 QLineEdit* QClient::Location [private]

6.72.4.14 QAction* QClient::LocationAction[LocationCount] [private]

6.72.4.15 const cardinal QClient::LocationCount = 15 [static, private]

6.72.4.16 const cardinal QClient::MaxLayerInfo = 3 [static, private]

6.72.4.17 const cardinal QClient::MaxScrollBarUpdateDelay = 4 [static, private]

6.72.4.18 QAction* QClient::MixerAction [private]

6.72.4.19 AudioMixer* QClient::MixerDevice [private]

6.72.4.20 QAudioMixer* QClient::MixerWindow [private]

6.72.4.21 QPushButton* QClient::Pause [private]

6.72.4.22 String QClient::PlayingURL [private]

6.72.4.23 QAction* QClient::ResolverAction [private]

6.72.4.24 QScrollBar* QClient::ScrollBar [private]

6.72.4.25 bool QClient::ScrollBarUpdated [private]

6.72.4.26 cardinal QClient::ScrollBarUpdateDelay [private]

6.72.4.27 QMenu* QClient::SettingsMenu [private]

6.72.4.28 QAction* QClient::SpectrumAnalyzerAction [private]

6.72.4.29 SpectrumAnalyzer* QClient::SpectrumAnalyzerDevice [private]

6.72.4.30 QSpectrumAnalyzer* QClient::SpectrumAnalyzerWindow [private]

6.72.4.31 QLabel* QClient::StatusBar [private]

6.72.4.32 QLabel* QClient::TitleLabel [private]

6.72.4.33 QMenu* QClient::ToolsMenu [private]

6.72.4.34 QList<String*> QClient::URLList [private]

6.72.4.35 QMenu* QClient::URLMenu [private]

6.72.4.36 QWhatsThis* QClient::WhatsThis [private]

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

- rtpa-qclient.h
- rtpa-qclient.cc

6.73 QInfoTabWidget Class Reference

QInfoTabWidget.

```
#include <qinfotabwidget.h>
```

Public Member Functions

- QInfoTabWidget (const InfoTable *table, const char *title, const char *pixmapName=NULL, QWidget *parent=NULL)

- QInfoWidget * addTable (const InfoTable *table, const char *title, const char *pixmapName=NULL)
- bool update (const QString &id, const QString &value)
- void clear ()

Private Attributes

- QList< QInfoWidget * > InfoWidgetList

6.73.1 Detailed Description

QInfoTabWidget.

This class is a widget for displaying groups of sets of info strings.

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

6.73.2 Constructor & Destructor Documentation

6.73.2.1 QInfoTabWidget::QInfoTabWidget (const InfoTable * *table*, const char * *title*, const char * *pixmapName* = NULL, QWidget * *parent* = NULL)

Constructor.

Parameters

table	InfoTable with info string descriptions.
title	Tab title.
pixmap-Name	Name of pixmap for tab.
parent	Parent widget.

6.73.3 Member Function Documentation

6.73.3.1 QInfoWidget * QInfoTabWidget::addTable (const InfoTable * *table*, const char * *title*, const char * *pixmapName* = NULL)

Add InfoTable to widget.

Parameters

table	InfoTable with info string descriptions.
title	Tab title.
pixmap-Name	Name of pixmap for tab.

6.73.3.2 void QInfoTabWidget::clear ( )

Clear all entries.

6.73.3.3 bool QInfoTabWidget::update (const QString & *id*, const QString & *value*)

Update string with given ID.

Parameters

id	String ID.
value	New value.

6.73.4 Member Data Documentation

6.73.4.1 QList<QInfoWidget*> QInfoTabWidget::InfoWidgetList [private]

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

- qinfotabwidget.h
- qinfotabwidget.cc

6.74 QInfoWidget Class Reference

QInfoWidget.

```
#include <qinfotabwidget.h>
```

Public Member Functions

- QInfoWidget (const InfoTable *table, QWidget *parent=NULL)
- bool update (const QString &id, const QString &value)
- void clear ()

Private Attributes

- const InfoTable * Table
- QHash< QString, QLabel * > LabelDict

6.74.1 Detailed Description

QInfoWidget.

This class is a widget for displaying sets of info strings.

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

6.74.2 Constructor & Destructor Documentation

6.74.2.1 QInfoWidget::QInfoWidget (const InfoTable * *table*, QWidget * *parent* = NULL)

Constructor.

Parameters

table	InfoTable with info string descriptions.
parent	Parent widget.

6.74.3 Member Function Documentation

6.74.3.1 void QInfoWidget::clear ( )

Clear all entries.

6.74.3.2 bool QInfoWidget::update (const QString & *id*, const QString & *value*)

Update string with given ID.

Parameters

id	String ID.
value	New value.

6.74.4 Member Data Documentation

6.74.4.1 QHash<QString,QLabel*> QInfoWidget::LabelDict [private]

6.74.4.2 const InfoTable* QInfoWidget::Table [private]

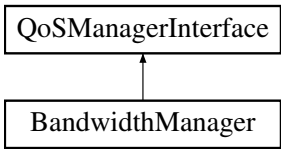
The documentation for this class was generated from the following files:

- qinfotabwidget.h
- qinfotabwidget.cc

6.75 QoSManagerInterface Class Reference

`#include <qosmanagerinterface.h>`

Inheritance diagram for QoSManagerInterface:



Public Member Functions

- virtual void addStream (ManagedStreamInterface *stream, const cardinal sessionID=0, const char *name=NULL)=0
- virtual void removeStream (ManagedStreamInterface *stream)=0
- virtual void updateStream (ManagedStreamInterface *stream)=0
- virtual void intervalChangeEvent (ManagedStreamInterface *stream, const bool isNew, const card64 when, const bool newRUList)=0
- virtual void reportEvent (ManagedStreamInterface *stream, const RTCP-ReceptionReportBlock *report, const cardinal layer)=0
- virtual void bufferFlushEvent (ManagedStreamInterface *stream, const cardinal layer)=0

6.75.1 Member Function Documentation

6.75.1.1 virtual void QoSManagerInterface::addStream (ManagedStreamInterface * *stream*, const cardinal *sessionID* = 0, const char * *name* = NULL) [pure virtual]

Add stream to management.

Parameters

stream	Stream to add.
session	SessionID of session to add stream to (0 for no session).
name	Stream name (only for log printing).

Implemented in BandwidthManager.

6.75.1.2 virtual void QoSManagerInterface::bufferFlushEvent (ManagedStreamInterface * *stream,* const cardinal *layer*) [pure virtual]

Buffer flush for a given layer.

Parameters

stream	Stream.

Implemented in BandwidthManager.

6.75.1.3 virtual void QoSManagerInterface::intervalChangeEvent (ManagedStreamInterface * *stream,* const bool *isNew,* const card64 *when,* const bool *newRUList*) [pure virtual]

Interval has changed.

Parameters

stream	Stream with changed interval.
isNew	true, if new interval has been reached; false otherwise.
when	Microseconds to next interval.
newRUList	true, if new resource/utiliztion list has been reached; false otherwise.

Implemented in BandwidthManager.

6.75.1.4 virtual void QoSManagerInterface::removeStream (ManagedStreamInterface * *stream*) [pure virtual]

Remove stream from management.

Parameters

stream	Stream to remove.

Implemented in BandwidthManager.

6.75.1.5 virtual void QoSManagerInterface::reportEvent (ManagedStreamInterface * *stream,* const RTCPReceptionReportBlock * *report,* const cardinal *layer*) [pure virtual]

Report reception for given layer.

Parameters

stream	Stream.
report	Report.
layer	Layer.

Implemented in BandwidthManager.

6.75.1.6 virtual void QoSManagerInterface::updateStream (ManagedStreamInterface * *stream*) [pure virtual]

Update stream.

Parameters

stream	Stream to be updated.

Implemented in BandwidthManager.

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

- qosmanagerinterface.h

6.76 QSpectrumAnalyzer Class Reference

QSpectrumAnalyzer.

```
#include <qspectrumanalyzer.h>
```

Public Slots

- void timerEvent ()
- void pause (bool on)
- void reset ()
- void closeWindow ()
- void newInterval (int index)
- void drawAverageLineToggled (int status)

Signals

- void closeSpectrumAnalyzer ()

Public Member Functions

- QSpectrumAnalyzer (SpectrumAnalyzer *analyzer, QWidget *parent=NULL)
- ~QSpectrumAnalyzer ()

Private Member Functions

- void closeEvent (QCloseEvent *event)

Private Attributes

- cardinal ArrayL [Bars]
- cardinal ArrayR [Bars]
- cardinal Max
- QSpectrumDisplay * PaintWidget1
- QSpectrumDisplay * PaintWidget2
- QCheckBox * Average
- QPushButton * Pause
- QTimer * Timer
- card16 Timing
- SpectrumAnalyzer * Analyzer

Static Private Attributes

- static const cardinal Bars = 70

6.76.1 Detailed Description

QSpectrumAnalyzer.

This class is the Qt-Toolkit GUI for the spectrum analyzer.

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

6.76.2 Constructor & Destructor Documentation

6.76.2.1 QSpectrumAnalyzer::QSpectrumAnalyzer (SpectrumAnalyzer * *analyzer*, QWidget * *parent* = NULL)

Constructor.

Parameters

analyzer	SpectrumAnalyzer object.
parent	Parent widget.

6.76.2.2 QSpectrumAnalyzer::~QSpectrumAnalyzer ( )

Destructor.

6.76.3 Member Function Documentation

6.76.3.1 void QSpectrumAnalyzer::closeEvent (QCloseEvent * *event*) [private]

6.76.3.2 void QSpectrumAnalyzer::closeSpectrumAnalyzer ( ) [signal]

Qt signal: Emitted, when "Close" or window's close button is clicked.

6.76.3.3 void QSpectrumAnalyzer::closeWindow ( ) [slot]

Qt slot: Close window.

6.76.3.4 void QSpectrumAnalyzer::drawAverageLineToggled (int *status*) [slot]

Qt slot: Change draw average line status.

6.76.3.5 void QSpectrumAnalyzer::newInterval (int *index*) [slot]

Qt slot: Change update interval.

6.76.3.6 void QSpectrumAnalyzer::pause (bool *on*) [slot]

Qt slot: Pause displaying the spectrum.

6.76.3.7 void QSpectrumAnalyzer::reset ( ) [slot]

Qt slot: Reset spectrum analyzer.

6.76.3.8 void QSpectrumAnalyzer::timerEvent ( ) [slot]

Qt slot: Called by QTimer.

6.76.4 Member Data Documentation

6.76.4.1 SpectrumAnalyzer* QSpectrumAnalyzer::Analyzer [private]

6.76.4.2 cardinal QSpectrumAnalyzer::ArrayL[Bars] [private]

6.76.4.3 cardinal QSpectrumAnalyzer::ArrayR[Bars] [private]

6.76.4.4 QCheckBox* QSpectrumAnalyzer::Average [private]

6.76.4.5 const cardinal QSpectrumAnalyzer::Bars = 70 [static, private]

6.76.4.6 cardinal QSpectrumAnalyzer::Max [private]

6.76.4.7 QSpectrumDisplay* QSpectrumAnalyzer::PaintWidget1 [private]

6.76.4.8 QSpectrumDisplay* QSpectrumAnalyzer::PaintWidget2 [private]

6.76.4.9 QPushButton* QSpectrumAnalyzer::Pause [private]

6.76.4.10 QTimer* QSpectrumAnalyzer::Timer [private]

6.76.4.11 card16 QSpectrumAnalyzer::Timing [private]

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

- qspectrumanalyzer.h
- qspectrumanalyzer.cc
- qspectrumanalyzer_moc.cc

6.77 QSpectrumDisplay Class Reference

QSpectrumAnalyzer.

`#include <qspectrumanalyzer.h>`

Public Slots

- void paintEvent (QPaintEvent *)
- void setDrawAverageLine (const bool drawAverageLine)

Public Member Functions

- QSpectrumDisplay (QWidget *parent, const cardinal *array, const cardinal bars, cardinal &max, const bool drawAverageLine=TRUE)
- ~QSpectrumDisplay ()

Private Member Functions

- void drawBar (QPainter *painter, const cardinal x, const cardinal y, const cardinal width, const cardinal height, const cardinal barValue)

Private Attributes

- const cardinal * Array
- cardinal Bars
- cardinal & Max
- bool DrawAverageLine

Static Private Attributes

- static const cardinal BarColors = 12
- static const cardinal AverageSteps = 10

6.77.1 Detailed Description

QSpectrumAnalyzer.

This class is the spectrum display widget for the spectrum analyzer.

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

6.77.2 Constructor & Destructor Documentation

6.77.2.1 QSpectrumDisplay::QSpectrumDisplay (QWidget * *parent*, const cardinal * *array*, const cardinal *bars*, cardinal & *max*, const bool *drawAverageLine* = TRUE)

Constructor.

Parameters

parent	Parent widget.
array	Fourier array.
bars	Number of fourier bars.
drawAverageLine	Draw (TRUE) or hide (FALSE) average line.

6.77.2.2 QSpectrumDisplay::~QSpectrumDisplay ( )

Destructor.

6.77.3 Member Function Documentation

6.77.3.1 void QSpectrumDisplay::drawBar (QPainter * *painter,* const cardinal *x,* const cardinal *y,* const cardinal *width,* const cardinal *height,* const cardinal *barValue*) [private]

6.77.3.2 void QSpectrumDisplay::paintEvent (QPaintEvent *) [slot]

Qt slot: Paint event.

6.77.3.3 void QSpectrumDisplay::setDrawAverageLine (const bool *drawAverageLine*) [inline, slot]

6.77.4 Member Data Documentation

6.77.4.1 const cardinal* QSpectrumDisplay::Array [private]

6.77.4.2 const cardinal QSpectrumDisplay::AverageSteps = 10 [static, private]

6.77.4.3 const cardinal QSpectrumDisplay::BarColors = 12 [static, private]

6.77.4.4 cardinal QSpectrumDisplay::Bars [private]

6.77.4.5 bool QSpectrumDisplay::DrawAverageLine [private]

6.77.4.6 cardinal& QSpectrumDisplay::Max [private]

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

- qspectrumanalyzer.h
- qspectrumanalyzer.cc

6.78 Randomizer Class Reference

Randomizer.

`#include <randomizer.h>`

Public Member Functions

- Randomizer ()
- void setSeed ()
- void setSeed (const cardinal seed)
- card8 random8 ()

- card16 random16 ()
- card32 random32 ()
- card64 random64 ()
- double random ()
- cardinal random (const cardinal a, const cardinal b)
- double random (const double a, const double b)

Private Attributes

- card32 Value

6.78.1 Detailed Description

Randomizer.

This class is an randomizer. The randomizer algorithm will calculate random numbers with seed given by system timer (microseconds since January 01, 1970) or given by a number.

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

6.78.2 Constructor & Destructor Documentation

6.78.2.1 Randomizer::Randomizer ( )

Constructor. Seed will be initialized by system timer (microseconds since January 01, 1970).

6.78.3 Member Function Documentation

6.78.3.1 double Randomizer::random ( ) [inline]

Get double random number out of interval [0,1].

Returns

The generated number.

6.78.3.2 cardinal Randomizer::random (const cardinal *a,* const cardinal *b*)

Get double random cardinal number out of interval [a,b].

Returns

The generated number.

6.78.3.3 double Randomizer::random (const double *a,* const double *b*)

Get double random double number out of interval [a,b].

Returns

The generated number.

6.78.3.4 card16 Randomizer::random16 ( ) [inline]

Get 16-bit random number.

Returns

The generated number.

6.78.3.5 card32 Randomizer::random32 ( ) [inline]

Get 32-bit random number.

Returns

The generated number.

6.78.3.6 card64 Randomizer::random64 ( ) [inline]

Get 64-bit random number.

Returns

The generated number.

6.78.3.7 card8 Randomizer::random8 ( ) [inline]

Get 8-bit random number.

Returns

The generated number.

6.78.3.8 void Randomizer::setSeed ( )

Set seed by system timer (microseconds since January 01, 1970).

6.78.3.9 void Randomizer::setSeed (const cardinal *seed*)

Set seed by given number.

Parameters

seed	Seed value.

6.78.4 Member Data Documentation

6.78.4.1 card32 Randomizer::Value [private]

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

- randomizer.h
- randomizer.cc

6.79 MultiAudioReader::ReaderEntry Struct Reference

Public Attributes

- AudioReaderInterface * Reader
- bool OverwriteSettings
- String Title
- String Artist
- String Comment

6.79.1 Member Data Documentation

6.79.1.1 String MultiAudioReader::ReaderEntry::Artist

6.79.1.2 String MultiAudioReader::ReaderEntry::Comment

6.79.1.3 bool MultiAudioReader::ReaderEntry::OverwriteSettings

6.79.1.4 AudioReaderInterface* MultiAudioReader::ReaderEntry::Reader

6.79.1.5 String MultiAudioReader::ReaderEntry::Title

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

- multiaudioreader.h

6.80 ResourceUtilizationMultiPoint Struct Reference

Resource Utilization Simple Point.

`#include <bandwidthmanager.h>`

Public Member Functions

- int operator< (const ResourceUtilizationMultiPoint &srup) const
- int operator> (const ResourceUtilizationMultiPoint &srup) const

Public Attributes

- SessionDescription * Session
- double SessionPriorityFactor
- cardinal Streams
- StreamDescription * Stream [MaxStreamsPerSession]
- cardinal Point [MaxStreamsPerSession]
- card64 Bandwidth
- double BandwidthCost
- double Utilization
- double SortingValue
- bool AlreadyAllocated

Static Public Attributes

- static const cardinal MaxStreamsPerSession = 128

6.80.1 Detailed Description

Resource Utilization Simple Point.

This is a resource/utilization multipoint structure to be used within the bandwidth manager.

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

6.80.2 Member Function Documentation

6.80.2.1 int ResourceUtilizationMultiPoint::operator< (const ResourceUtilizationMultiPoint & *srup*) const [inline]

Operator "<".

6.80.2.2 int ResourceUtilizationMultiPoint::operator> (const ResourceUtilizationMultiPoint & *srup*) const [inline]

Operator "<".

6.80.3 Member Data Documentation

6.80.3.1 bool ResourceUtilizationMultiPoint::AlreadyAllocated

True, if this point has already been allocated during session's minimum bandwidth allocation.

6.80.3.2 card64 ResourceUtilizationMultiPoint::Bandwidth

Bandwidth.

6.80.3.3 double ResourceUtilizationMultiPoint::BandwidthCost

Bandwidth cost.

6.80.3.4 const cardinal ResourceUtilizationMultiPoint::MaxStreamsPerSession = 128 [static]

Maximum number of streams per session.

6.80.3.5 cardinal ResourceUtilizationMultiPoint::Point[MaxStreamsPerSession]

Array of point numbers for this multipoint's points.

6.80.3.6 SessionDescription* ResourceUtilizationMultiPoint::Session

SessionDescription of this multipoint's session.

6.80.3.7 double ResourceUtilizationMultiPoint::SessionPriorityFactor

Session's priority factor.

6.80.3.8 double ResourceUtilizationMultiPoint::SortingValue

Sorting value.

6.80.3.9 StreamDescription* ResourceUtilizationMultiPoint::Stream[MaxStreamsPerSession]

Array of StreamDescriptions for this multipoint's streams.

6.80.3.10 cardinal ResourceUtilizationMultiPoint::Streams

Number of streams in this session.

6.80.3.11 double ResourceUtilizationMultiPoint::Utilization

Utilization.

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

- bandwidthmanager.h

6.81 ResourceUtilizationPoint Class Reference

Resource Utilization Point.

`#include <resourceutilizationpoint.h>`

Public Member Functions

- void reset ()
- int operator== (const ResourceUtilizationPoint &rup) const
- int operator!= (const ResourceUtilizationPoint &rup) const

Static Public Member Functions

- static cardinal mergeResourceUtilizationLists (ResourceUtilizationPoint *destination, ResourceUtilizationPoint **listArray, const cardinal *listSizeArray, const cardinal listCount)
- static void sortResourceUtilizationList (ResourceUtilizationPoint *rup, const integer start, const integer end)
- static cardinal optimizeResourceUtilizationList (ResourceUtilizationPoint *rup, const cardinal count)
- static cardinal grahamScanResourceUtilizationList (ResourceUtilizationPoint *rup, const cardinal count)

Public Attributes

- card64 Bandwidth
- double BandwidthCost
- double Utilization
- double FrameRate
- cardinal Layers
- BandwidthInfo LayerBandwidthInfo [RTPConstants::RTPMaxQualityLayers]
- LayerClassMapping Mapping [RTPConstants::RTPMaxQualityLayers]

Static Private Member Functions

- static void swapResourceUtilizationPoints (ResourceUtilizationPoint &a, ResourceUtilizationPoint &b)
- static integer ccw (const ResourceUtilizationPoint &p0, const ResourceUtilizationPoint &p1, const ResourceUtilizationPoint &p2)

6.81.1 Detailed Description

Resource Utilization Point.

This class is a resource/utilization point used for the bandwidth mapping algorithm.

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

6.81.2 Member Function Documentation

6.81.2.1 static integer ResourceUtilizationPoint::ccw (const ResourceUtilizationPoint & *p0*, const ResourceUtilizationPoint & *p1*, const ResourceUtilizationPoint & *p2*) [inline, static, private]

6.81.2.2 cardinal ResourceUtilizationPoint::grahamScanResourceUtilizationList (ResourceUtilizationPoint * *rup*, const cardinal *count*) [static]

Compute convex hull on resource/utilization list using Graham Scan algorithm.

Parameters

rup	List.
count	Number of entries in list.

Returns

Number of entries remaining in list.

6.81.2.3 cardinal ResourceUtilizationPoint::mergeResourceUtilizationLists (ResourceUtilizationPoint * *destination*, ResourceUtilizationPoint ** *listArray*, const cardinal * *listSizeArray*, const cardinal *listCount*) [static]

Merge resource/utilization lists.

Parameters

destination	Destination list.
listArray	Array of lists to merge.
listSizeArray	Array of list sizes.
listCount	Number of lists.

Returns

Number of points in destination list.

6.81.2.4 int ResourceUtilizationPoint::operator!= (const ResourceUtilizationPoint & *rup*) const [inline]

Operator "!=".

6.81.2.5 int ResourceUtilizationPoint::operator== (const ResourceUtilizationPoint & *rup*) const [inline]

Operator "==".

6.81.2.6 cardinal ResourceUtilizationPoint::optimizeResourceUtilizationList (ResourceUtilizationPoint * *rup*, const cardinal *count*) [static]

Optimize resource/utilization list by utilization: Eliminate points which have higher resource requirements or cost than higher-utilized points following.

Parameters

rup	List.
count	Number of entries in list.

Returns

Number of entries remaining in list.

6.81.2.7 void ResourceUtilizationPoint::reset ( )

Reset.

6.81.2.8 void ResourceUtilizationPoint::sortResourceUtilizationList (ResourceUtilizationPoint * *rup,* const integer *start,* const integer *end*) [static]

Sort resource/utilization list by utilization.

Parameters

rup	List.
start	First point number.
end	Last point number.

6.81.2.9 static void ResourceUtilizationPoint::swapResourceUtilizationPoints (ResourceUtilizationPoint & *a,* ResourceUtilizationPoint & *b*) [inline, static, private]

6.81.3 Member Data Documentation

6.81.3.1 card64 ResourceUtilizationPoint::Bandwidth

Total bandwidth.

6.81.3.2 double ResourceUtilizationPoint::BandwidthCost

Bandwidth cost.

6.81.3.3 double ResourceUtilizationPoint::FrameRate

Frame rate.

6.81.3.4 BandwidthInfo ResourceUtilizationPoint::LayerBandwidthInfo[RTPConstants::RTPMaxQualityLayers]

Array of layers' bandwidth requirements.

6.81.3.5 cardinal ResourceUtilizationPoint::Layers

Number of layers.

6.81.3.6 LayerClassMapping ResourceUtilizationPoint::Mapping[RTPConstants::-RTPMaxQualityLayers]

Layer to DiffServ class mapping possibilities.

6.81.3.7 double ResourceUtilizationPoint::Utilization

Total utilization.

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

- resourceutilizationpoint.h
- resourceutilizationpoint.cc

6.82 ResourceUtilizationSimplePoint Struct Reference

Resource Utilization Simple Point.

`#include <bandwidthmanager.h>`

Public Attributes

- StreamDescription * Stream
- cardinal Point
- double StreamPriorityFactor
- card64 Bandwidth
- double BandwidthCost
- double Utilization
- double SortingValue

6.82.1 Detailed Description

Resource Utilization Simple Point.

This is a resource/utilization point structure to be used within the bandwidth manager.

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

6.82.2 Member Data Documentation

6.82.2.1 card64 ResourceUtilizationSimplePoint::Bandwidth

Bandwidth.

6.82.2.2 double ResourceUtilizationSimplePoint::BandwidthCost

Bandwidth cost.

6.82.2.3 cardinal ResourceUtilizationSimplePoint::Point

Point number (StreamDescription's RUList entry number).

6.82.2.4 double ResourceUtilizationSimplePoint::SortingValue

Sorting value.

6.82.2.5 StreamDescription* ResourceUtilizationSimplePoint::Stream

StreamDescription of this point's stream.

6.82.2.6 double ResourceUtilizationSimplePoint::StreamPriorityFactor

Stream's priority factor.

6.82.2.7 double ResourceUtilizationSimplePoint::Utilization

Utilization.

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

- bandwidthmanager.h

6.83 WavAudioReader::RIFF_Chunk Struct Reference

Public Attributes

- char ID [4]
- card32 Length

6.83.1 Member Data Documentation

6.83.1.1 char WavAudioReader::RIFF_Chunk::ID[4]

6.83.1.2 card32 WavAudioReader::RIFF_Chunk::Length

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

- wavaudioreader.h

6.84 WavAudioReader::RIFF_Header Struct Reference

Public Attributes

- char RIFF [4]
- card32 Length
- char FormatID [4]

6.84.1 Member Data Documentation

6.84.1.1 char WavAudioReader::RIFF_Header::FormatID[4]

6.84.1.2 card32 WavAudioReader::RIFF_Header::Length

6.84.1.3 char WavAudioReader::RIFF_Header::RIFF[4]

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

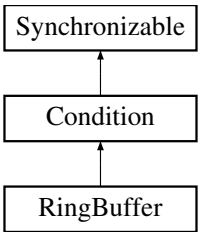
- wavaudioreader.h

6.85 RingBuffer Class Reference

Ring Buffer.

`#include <ringbuffer.h>`

Inheritance diagram for RingBuffer:



Public Member Functions

- RingBuffer ()
- ~RingBuffer ()
- bool init (const cardinal bytes)
- void flush ()
- size_t bytesReadable ()
- size_t bytesWritable ()
- ssize_t read (char *data, const size_t length)
- ssize_t write (const char *data, const size_t length)

Private Attributes

- char * Buffer
- size_t BufferSize
- size_t WriteStart
- size_t WriteEnd
- size_t BytesStored

6.85.1 Detailed Description

Ring Buffer.

This class implements a ring buffer.

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

6.85.2 Constructor & Destructor Documentation

6.85.2.1 RingBuffer::RingBuffer ( )

Constructor.

6.85.2.2 RingBuffer::~RingBuffer ( )

Destructor.

6.85.3 Member Function Documentation

6.85.3.1 size_t RingBuffer::bytesReadable ( ) [inline]

Get number of bytes available for read.

Returns

Number of bytes readable.

6.85.3.2 size_t RingBuffer::bytesWritable ( ) [inline]

Get number of bytes available for write = (BufferSize - bytesReadable()).

Returns

Number of bytes writable.

6.85.3.3 void RingBuffer::flush ( )

Flush buffer.

6.85.3.4 bool RingBuffer::init (const cardinal *bytes*)

Initialize ring buffer.

Parameters

bytes	Number of bytes to allocate for buffer.

Returns

true for success; false otherwise.

6.85.3.5 ssize_t RingBuffer::read (char * *data*, const size_t *length*)

Read data from ring buffer.

Parameters

data	Data buffer to store read data to.
length	Size of data buffer.

Returns

Bytes read from ring buffer.

6.85.3.6 ssize_t RingBuffer::write (const char * *data*, const size_t *length*)

Write data into ring buffer.

Parameters

data	Data buffer containing data to write.
length	Length of data to write.

Returns

Bytes written into ring buffer.

6.85.4 Member Data Documentation

6.85.4.1 char* RingBuffer::Buffer [private]

6.85.4.2 size_t RingBuffer::BufferSize [private]

6.85.4.3 size_t RingBuffer::BytesStored [private]

6.85.4.4 size_t RingBuffer::WriteEnd [private]

6.85.4.5 size_t RingBuffer::WriteStart [private]

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

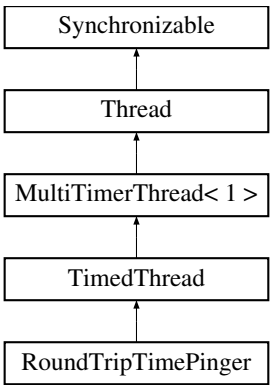
- ringbuffer.h
- ringbuffer.cc

6.86 RoundTripTimePinger Class Reference

Round Trip Time Pinger.

`#include <roundtriptimepinger.h>`

Inheritance diagram for RoundTripTimePinger:



Classes

- struct Ping4Packet
- struct Ping6Packet

Public Member Functions

- RoundTripTimePinger (Socket *ping4socket, Socket *ping6socket, const card64 delay=1000000)
- ~RoundTripTimePinger ()
- bool ready () const
- cardinal getHosts ()
- double getAlpha ()
- void setAlpha (const double alpha)
- card64 getMaxPingDelay ()
- void setMaxPingDelay (const card64 delay)
- cardinal getRoundTripTime (const InternetAddress &address, const card8 trafficClass=0x00)
- bool addHost (const InternetAddress &address, const card8 trafficClass=0x00)
- void removeHost (const InternetAddress &address, const card8 trafficClass=0x00)
- void activateLogger (std::ostream *scriptStream, std::ostream *dataStream, const char *dataName)
- void deactivateLogger ()
- bool isLogging () const
- void writeGPHeader (std::ostream &os, const char *dataName, const cardinal lineStyle=1)
- void writeGPData (std::ostream &os)

Static Public Attributes

- static const cardinal MaxRoundTripTime = 180000000
- static const double UnreachableFactor = 2.0
- static const card64 MinUnreachableAsumption = 2500000

Private Member Functions

- void timerEvent ()
- void calculateRoundTripTime (const InternetAddress &address, const card8 trafficClass, const card64 sendTime, const card64 arrivalTime)
- card16 calculateChecksum (const card16 *addr, const cardinal length, card16 csum)
- card64 sendPing4 (const InternetAddress &destination, const card8 trafficClass, const card16 sequenceNumber)
- card64 sendPing6 (const InternetAddress &destination, const card8 trafficClass, const card16 sequenceNumber)
- bool receiveEcho4 ()
- bool receiveEcho6 ()
- void checkUnreachable (PingerHost &host)

Private Attributes

- Socket * Ping4Socket
- Socket * Ping6Socket
- double RoundTripTimeAlpha
- std::multiset< PingerHost > HostSet
- card64 GPHeaderTimeStamp
- bool Ready
- bool Logger
- std::ostream * LoggerScriptStream
- std::ostream * LoggerDataStream
- card64 MaxPingDelay
- Randomizer Random

Friends

- std::ostream & operator<< (std::ostream &os, RoundTripTimePinger &pinger)

6.86.1 Detailed Description

Round Trip Time Pinger.

This class implements a round trip time pinger.

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

6.86.2 Constructor & Destructor Documentation

6.86.2.1 RoundTripTimePinger::RoundTripTimePinger (Socket * *ping4socket*, Socket * *ping6socket*, const card64 *delay* = 1000000)

Constructor.

Parameters

ping4socket	Socket for IPv4 pings.
ping6socket	Socket for IPv6 pings.
delay	Maximum delay between two pings in microseconds.

6.86.2.2 RoundTripTimePinger::~RoundTripTimePinger ( )

Destructor.

6.86.3 Member Function Documentation

6.86.3.1 void RoundTripTimePinger::activateLogger (std::ostream * *scriptStream*, std::ostream * *dataStream*, const char * *dataName*)

Activate logger. Very important: Logging will be deactivated by addHost() and removeHost() calls!

Parameters

scriptStream	Script output stream.
dataStream	Data output stream.
dataName	Data file name (for GNUplot's plot command).

6.86.3.2 bool RoundTripTimePinger::addHost (const InternetAddress & *address*, const card8 *trafficClass* = 0x00)

Add host to RoundTripTimePinger list.

Parameters

address	Host address.
trafficClass	Traffic class.

Returns

true, if host has been added; false otherwise (duplicate).

6.86.3.3 card16 RoundTripTimePinger::calculateChecksum (const card16 * *addr,* const cardinal *length,* card16 *csum*) [private]

6.86.3.4 void RoundTripTimePinger::calculateRoundTripTime (const InternetAddress & *address,* const card8 *trafficClass,* const card64 *sendTime,* const card64 *arrivalTime*) [private]

6.86.3.5 void RoundTripTimePinger::checkUnreachable (PingerHost & *host*) [private]

6.86.3.6 void RoundTripTimePinger::deactivateLogger ( )

Deactivate logger.

6.86.3.7 double RoundTripTimePinger::getAlpha ( ) [inline]

Get constant alpha: RTT = alpha * oldValue + (1 - alpha) * newValue.

Returns

alpha.

6.86.3.8 cardinal RoundTripTimePinger::getHosts ( ) [inline]

Get number of hosts in RoundTripTimePinger.

Returns

Number of hosts.

6.86.3.9 card64 RoundTripTimePinger::getMaxPingDelay ( ) [inline]

Get maximum delay between two pings in microseconds.

Returns

Delay in microseconds.

6.86.3.10 cardinal RoundTripTimePinger::getRoundTripTime (const InternetAddress & *address,* const card8 *trafficClass* = 0x00)

Get round trip time for given host and traffic class.

Parameters

trafficClass	Traffic class.

Returns

Round trip time in microseconds; -1 for hosts not in list or unreachable.

6.86.3.11 bool RoundTripTimePinger::isLogging ( ) const [inline]

Check, if logger is running.

6.86.3.12 bool RoundTripTimePinger::ready ( ) const [inline]

Check, if RoundTripTimePinger is ready.

Returns

true, if RoundTripTimePinger is ready; false otherwise.

6.86.3.13 bool RoundTripTimePinger::receiveEcho4 ( ) [private]

6.86.3.14 bool RoundTripTimePinger::receiveEcho6 ( ) [private]

6.86.3.15 void RoundTripTimePinger::removeHost (const InternetAddress & *address,* const card8 *trafficClass* = 0x00)

Remove host from RoundTripTimePinger list.

Parameters

address	Host address.
trafficClass	Traffic class.

6.86.3.16 card64 RoundTripTimePinger::sendPing4 (const InternetAddress & *destination,* const card8 *trafficClass,* const card16 *sequenceNumber*) [private]

6.86.3.17 card64 RoundTripTimePinger::sendPing6 (const InternetAddress & *destination,* const card8 *trafficClass,* const card16 *sequenceNumber*) [private]

6.86.3.18 void RoundTripTimePinger::setAlpha (const double *alpha*) [inline]

Set constant alpha: RTT = alpha ∗ oldValue + (1 - alpha) ∗ newValue.

Parameters

alpha	Alpha.

6.86.3.19 void RoundTripTimePinger::setMaxPingDelay (const card64 *delay*) [inline]

Set maximum delay between two pings in microseconds.

Parameters

delay	Delay in microseconds.

6.86.3.20 void RoundTripTimePinger::timerEvent ( ) [private, virtual]

The virtual timerEvent() method, which contains the timed thread's implementation. It has to be implemented by classes, which inherit TimedThread. This method is called regularly with the given interval.

Implements TimedThread.

6.86.3.21 void RoundTripTimePinger::writeGPData (std::ostream & *os*)

Write GNUplot data line.

Parameters

os	Output stream.

6.86.3.22 void RoundTripTimePinger::writeGPHeader (std::ostream & *os,* const char ∗ *dataName,* const cardinal *lineStyle* = 1)

Write GNUplot header. Very important: This header will become invalid when calling addHost() or removeHost()!

Parameters

os	Output stream.
dataName	Name of data file.
lineStyle	First GNUplot line style or 0 for using GNUplot's defaults.

6.86.4 Friends And Related Function Documentation

6.86.4.1 std::ostream& operator<< (std::ostream & *os,* RoundTripTimePinger & *pinger*) [friend]

Friend output operator.

6.86.5 Member Data Documentation

6.86.5.1 card64 RoundTripTimePinger::GPHeaderTimeStamp [private]

6.86.5.2 std::multiset<PingerHost> RoundTripTimePinger::HostSet [private]

6.86.5.3 bool RoundTripTimePinger::Logger [private]

6.86.5.4 std::ostream* RoundTripTimePinger::LoggerDataStream [private]

6.86.5.5 std::ostream* RoundTripTimePinger::LoggerScriptStream [private]

6.86.5.6 card64 RoundTripTimePinger::MaxPingDelay [private]

6.86.5.7 const cardinal RoundTripTimePinger::MaxRoundTripTime = 180000000 [static]

Maximum round trip time in microseconds.

6.86.5.8 const card64 RoundTripTimePinger::MinUnreachableAsumption = 2500000 [static]

MinUnreachableAsumption: Assume current round trip time to be diff = now - host.-LastEchoTimeStamp, if diff > MinUnreachableAsumption (for OS delay) or diff > -UnreachableFactor * MaxRawRoundTripTime (for real network delay).

6.86.5.9 Socket* RoundTripTimePinger::Ping4Socket [private]

6.86.5.10 Socket* RoundTripTimePinger::Ping6Socket [private]

6.86.5.11 Randomizer RoundTripTimePinger::Random [private]

6.86.5.12 bool RoundTripTimePinger::Ready [private]

6.86.5.13 double RoundTripTimePinger::RoundTripTimeAlpha [private]

6.86.5.14 const double RoundTripTimePinger::UnreachableFactor = 2.0 [static]

Unreachable Factor: Assume current round trip time to be diff = now - host.LastEcho-TimeStamp, if diff $>$ MinUnreachableAsumption (for OS delay) or diff $>$ Unreachable-Factor $*$ MaxRawRoundTripTime (for real network delay).

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

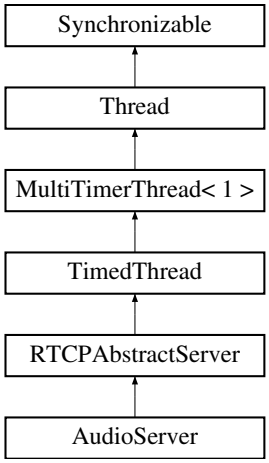
- roundtriptimepinger.h
- roundtriptimepinger.cc

6.87 RTCPAbstractServer Class Reference

RTCP abstract server.

`#include <rtcpabstractserver.h>`

Inheritance diagram for RTCPAbstractServer:



Classes

- struct Client

Public Types

- enum DeleteReason { DeleteReason_UserBye = 0, DeleteReason_Timeout = 1, DeleteReason_Shutdown = 2, DeleteReason_Error = 3 }

Public Member Functions

- RTCPAbstractServer ()
- ~RTCPAbstractServer ()
- virtual void * newClient (Client *client, const char *cname)=0
- virtual void deleteClient (Client *client, const DeleteReason reason)=0
- virtual bool checkClient (Client *client)=0
- virtual void appMessage (Client *client, const char *name, void *data, const cardinal dataLength)=0
- virtual void sdesMessage (Client *client, const card8 type, char *data, const cardinal length)=0
- virtual void receiverReport (Client *client, RTCPReceptionReportBlock *report, const cardinal layer)=0
- virtual void outOfMemoryWarning ()
- cardinal getMembers ()
- card64 getDefaultTimeout () const
- void setDefaultTimeout (const card64 timeout)
- void * stop ()

Private Member Functions

- void receivedSenderReport (const InternetFlow flow, const card32 source, RTCPReceptionReportBlock *report, const cardinal layer)
- void receivedReceiverReport (const InternetFlow flow, const card32 source, RTCPReceptionReportBlock *report, const cardinal layer)
- void receivedSourceDescription (const InternetFlow flow, const card32 source, const card8 type, char *data, const card8 length)
- void receivedApp (const InternetFlow flow, const card32 source, const char *name, void *data, const card32 dataLength)
- void receivedBye (const InternetFlow flow, const card32 source, const DeleteReason reason)
- Client * findClient (const card32 source, const InternetFlow flow)
- void timerEvent ()

Private Attributes

- card64 DefaultTimeout
- std::multimap< const cardinal, Client * > ClientSet

Friends

- class RTCPReceiver

6.87.1 Detailed Description

RTCP abstract server.

This class is an abstract RTCP server.

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

6.87.2 Member Enumeration Documentation

6.87.2.1 enum RTCPAbstractServer::DeleteReason

A set of reasons for deleteClient() call.

Enumerator:

DeleteReason_UserBye

DeleteReason_Timeout

DeleteReason_Shutdown

DeleteReason_Error

6.87.3 Constructor & Destructor Documentation

6.87.3.1 RTCPAbstractServer::RTCPAbstractServer ( )

Constructor.

6.87.3.2 RTCPAbstractServer::~RTCPAbstractServer ( )

Destructor.

6.87.4 Member Function Documentation

6.87.4.1 virtual void RTCPAbstractServer::appMessage (Client * *client*, const char * *name*, void * *data*, const cardinal *dataLength*) [pure virtual]

Called when a client sends RTCP APP message. The call is synchronized by RTCPAbstractServer.

Parameters

client	Client.
name	RTCP APP name.
data	RTCP APP data.
dataLength	RTCP APP data length.

Implemented in AudioServer.

6.87.4.2 virtual bool RTCPAbstractServer::checkClient (Client * *client*) [pure virtual]

This method is called about once per second to check, if the client is okay (e.g. no transmission error has occurred etc.) The call is synchronized by RTCPAbstractServer.

Returns

true, if client is okay; false to delete client in case of an error.

Implemented in AudioServer.

6.87.4.3 virtual void RTCPAbstractServer::deleteClient (Client * *client,* const DeleteReason *reason*) [pure virtual]

Called when a client sends RTCP BYE or the timeout is reached. The call is synchronized by RTCPAbstractServer.

Parameters

client	Client.
reason	Reason for deleteClient() call.
hasTimeout	true, if timeout is reached; false, if RTCP BYE received.
shutdown	true, if server shutdown is in progress.

Implemented in AudioServer.

6.87.4.4 RTCPAbstractServer::Client * RTCPAbstractServer::findClient (const card32 *source,* const InternetFlow *flow*) [private]

6.87.4.5 card64 RTCPAbstractServer::getDefaultTimeout ( ) const [inline]

Get the default timeout in microseconds, after which a client is assumed to be dead and removed.

Returns

Default timeout in microseconds.

6.87.4.6 cardinal RTCPAbstractServer::getMembers ( ) [inline]

Get number of members serverd by the server.

Returns

Number of members.

6.87.4.7 virtual void* RTCPAbstractServer::newClient (Client * *client*, const char * *cname*) [pure virtual]

Called when a new client sends its SDES CNAME message. The class inheriting RTCPAbstractServer may use the client->UserData field to store additional data to serve the client. The result of the call will be saved into this field (client->UserData = newClient(client))! The call is synchronized by RTCPAbstractServer.

Parameters

client	Client.
cname	CNAME string.

Returns

A value, which RTCPAbstractServer will save to client->UserData.

Implemented in AudioServer.

6.87.4.8 void RTCPAbstractServer::outOfMemoryWarning ( ) [virtual]

This method is called, if an out of memory error occurs. It prints a simple error message. It should be overloaded by a more useful method within the concrete server. The call is synchronized by RTCPAbstractServer.

Reimplemented in AudioServer.

6.87.4.9 void RTCPAbstractServer::receivedApp (const InternetFlow *flow*, const card32 *source*, const char * *name*, void * *data*, const card32 *dataLength*) [private]

6.87.4.10 void RTCPAbstractServer::receivedBye (const InternetFlow *flow*, const card32 *source*, const DeleteReason *reason*) [private]

6.87.4.11 void RTCPAbstractServer::receivedReceiverReport (const InternetFlow *flow*, const card32 *source*, RTCPReceptionReportBlock * *report*, const cardinal *layer*) [private]

6.87.4.12 void RTCPAbstractServer::receivedSenderReport (const InternetFlow *flow,* const card32 *source,* RTCPReceptionReportBlock * *report,* const cardinal *layer*) [private]

6.87.4.13 void RTCPAbstractServer::receivedSourceDescription (const InternetFlow *flow,* const card32 *source,* const card8 *type,* char * *data,* const card8 *length*) [private]

6.87.4.14 virtual void RTCPAbstractServer::receiverReport (Client * *client,* RTCPReceptionReportBlock * *report,* const cardinal *layer*) [pure virtual]

Called when a client sends a receiver report; it is called for every receiver report block in the message. The call is synchronized by RTCPAbstractServer.

Parameters

client	Client.
report	RTCPReceptionReportBlock.
layer	Layer number.

Implemented in AudioServer.

6.87.4.15 virtual void RTCPAbstractServer::sdesMessage (Client * *client,* const card8 *type,* char * *data,* const cardinal *length*) [pure virtual]

Called when a client sends RTCP SDES message; it is called for every SDES item in the message. The call is synchronized by RTCPAbstractServer.

Parameters

client	Client.
type	RTCP SDES type.
data	RTCP SDES data.
length	RTCP SDES length.

Implemented in AudioServer.

6.87.4.16 void RTCPAbstractServer::setDefaultTimeout (const card64 *timeout*) [inline]

Set the default timeout in microseconds, after which a client is assumed to be dead and removed. The new value will be used for all new clients. Timeouts of old clients are not changed!

Parameters

timeout	Default timeout in microseconds.

6.87.4.17 void * RTCPAbstractServer::stop ( ) [virtual]

Reimplementation of Thread::stop() to remove all clients before stopping.

See also

Thread::stop

Reimplemented from MultiTimerThread< Timers >.

6.87.4.18 void RTCPAbstractServer::timerEvent ( ) [private, virtual]

The virtual timerEvent() method, which contains the timed thread's implementation. It has to be implemented by classes, which inherit TimedThread. This method is called regularly with the given interval.

Implements TimedThread.

6.87.5 Friends And Related Function Documentation

6.87.5.1 friend class RTCPReceiver [friend]

RTCPReceiver is friend class to enable usage of receivedXXX() methods.

6.87.6 Member Data Documentation

6.87.6.1 std::multimap<const cardinal,Client*> RTCPAbstractServer::ClientSet [private]

6.87.6.2 card64 RTCPAbstractServer::DefaultTimeout [private]

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

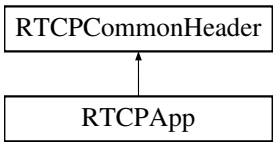
- rtcpabstractserver.h
- rtcpabstractserver.cc

6.88 RTCPApp Struct Reference

RTCP APP Message.

`#include <rtcppacket.h>`

Inheritance diagram for RTCPApp:



Public Member Functions

- RTCPApp ()
- RTCPApp (const card8 subtype)
- void init (const card8 subtype)
- card32 getSource () const
- char * getName ()
- char * getData ()
- void setSource (const card32 source)
- void setName (const char *name)

Private Attributes

- card32 Source
- char Name [4]
- char Data []

6.88.1 Detailed Description

RTCP APP Message.

This struct manages an RTCP APP message

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

See also

RTCPSender
RTCPReceiver
RTCPAbstractServer

6.88.2 Constructor & Destructor Documentation

6.88.2.1 RTCPApp::RTCPApp ( )

Constructor.

6.88.2.2 RTCPApp::RTCPApp (const card8 *subtype*)

Constructor.

Parameters

subtype	RTCP APP subtype.

6.88.3 Member Function Documentation

6.88.3.1 char* RTCPApp::getData ( ) [inline]

Get pointer to data field.

Returns

Pointer to data field.

6.88.3.2 char* RTCPApp::getName ( ) [inline]

Get pointer to name field.

Returns

Pointer to name field.

6.88.3.3 card32 RTCPApp::getSource ( ) const [inline]

Get source.

Returns

Source.

6.88.3.4 void RTCPApp::init (const card8 *subtype*)

6.88.3.5 void RTCPApp::setName (const char * *name*) [inline]

Set name.

Returns

Pointer to name field.

6.88.3.6 void RTCPApp::setSource (const card32 *source*) [inline]

Set source.

Parameters

source	Source.

6.88.4 Member Data Documentation

6.88.4.1 char RTCPApp::Data[] [private]

6.88.4.2 char RTCPApp::Name[4] [private]

6.88.4.3 card32 RTCPApp::Source [private]

The documentation for this struct was generated from the following files:

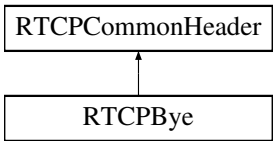
- rtcppacket.h
- rtcppacket.cc

6.89 RTCPBye Struct Reference

RTCP BYE Message.

`#include <rtcppacket.h>`

Inheritance diagram for RTCPBye:



Public Member Functions

- RTCPBye ()
- RTCPBye (const card8 count)
- void init (const card8 count)
- card32 getSource (const cardinal index) const
- void setSource (const cardinal index, const card32 source)

Private Attributes

- card32 Source [ ]

6.89.1 Detailed Description

RTCP BYE Message.

This struct manages an RTCP BYE message

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

See also

RTCPSender
RTCPReceiver
RTCPAbstractServer

6.89.2 Constructor & Destructor Documentation

6.89.2.1 RTCPBye::RTCPBye ( )

Constructor.

6.89.2.2 RTCPBye::RTCPBye (const card8 *count*)

Constructor.

Parameters

Count	count.

6.89.3 Member Function Documentation

6.89.3.1 card32 RTCPBye::getSource (const cardinal *index*) const [inline]

Get source at given index.

Parameters

index	Index.

Returns

Source.

6.89.3.2 void RTCPBye::init (const card8 *count*)

Initialize.

Parameters

Count	count.

6.89.3.3 void RTCPBye::setSource (const cardinal *index*, const card32 *source*) [inline]

Set source at given index.

Parameters

index	Index.
source	Source.

6.89.4 Member Data Documentation

6.89.4.1 card32 RTCPBye::Source[] [private]

The documentation for this struct was generated from the following files:

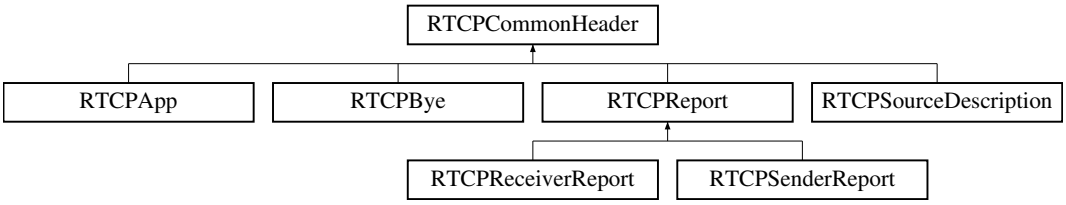
- rtcppacket.h
- rtcppacket.cc

6.90 RTCPCommonHeader Struct Reference

RTCP Common Header.

`#include <rtcppacket.h>`

Inheritance diagram for RTCPCommonHeader:



Public Member Functions

- RTCPCommonHeader ()
- card8 getVersion () const
- card8 getPadding () const
- card8 getCount () const
- card8 getPacketType () const
- card16 getLength () const
- void setVersion (const card8 version)
- void setPadding (const card8 padding)
- void setCount (const card8 count)
- void setPacketType (const card8 packetType)
- void setLength (const card16 length)

Protected Attributes

- card8 V:2
- card8 P:1
- card8 C:5
- card8 PT:8
- card16 Length

6.90.1 Detailed Description

RTCP Common Header.

This struct manages a common RTCP header.

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

See also

RTCPSender
RTCPReceiver
RTCPAbstractServer

6.90.2 Constructor & Destructor Documentation

6.90.2.1 RTCPCommonHeader::RTCPCommonHeader ( )

Constructor.

6.90.3 Member Function Documentation

6.90.3.1 card8 RTCPCommonHeader::getCount ( ) const [inline]

Get count.

Returns

RTCP count.

6.90.3.2 card16 RTCPCommonHeader::getLength ( ) const [inline]

Get length.

Returns

RTCP Length.

6.90.3.3 card8 RTCPCommonHeader::getPacketType ( ) const [inline]

Get packet type.

Returns

RTCP packet type.

6.90.3.4 card8 RTCPCommonHeader::getPadding ( ) const [inline]

Get padding.

Returns

RTCP padding.

6.90.3.5 card8 RTCPCommonHeader::getVersion ( ) const [inline]

Get version.

Returns

RTCP version.

6.90.3.6 void RTCPCommonHeader::setCount (const card8 *count*) [inline]

Set count.

Parameters

count	RTCP count.

6.90.3.7 void RTCPCommonHeader::setLength (const card16 *length*) [inline]

Set length.

Parameters

length	RTCP Length.

6.90.3.8 void RTCPCommonHeader::setPacketType (const card8 *packetType*) [inline]

Set packetType.

Parameters

packetType	RTCP packet Type.

6.90.3.9 void RTCPCommonHeader::setPadding (const card8 *padding*) [inline]

Set padding.

Parameters

padding	RTCP padding.

6.90.3.10 void RTCPCommonHeader::setVersion (const card8 *version*) [inline]

Set version.

Parameters

version	RTCP version.

6.90.4 Member Data Documentation

6.90.4.1 card8 RTCPCommonHeader::C [protected]

6.90.4.2 card16 RTCPCommonHeader::Length [protected]

6.90.4.3 card8 RTCPCommonHeader::P [protected]

6.90.4.4 card8 RTCPCommonHeader::PT [protected]

6.90.4.5 card8 RTCPCommonHeader::V [protected]

The documentation for this struct was generated from the following files:

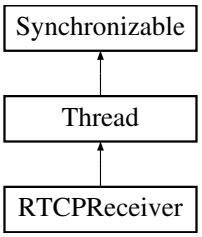
- rtcppacket.h
- rtcppacket.cc

6.91 RTCPReceiver Class Reference

RTCP Receiver.

`#include <rtcpreceiver.h>`

Inheritance diagram for RTCPReceiver:



Public Member Functions

- RTCPReceiver ()
- RTCPReceiver (RTCPAbstractServer *server, Socket *receiverSocket)
- ~RTCPReceiver ()
- void init (RTCPAbstractServer *server, Socket *receiverSocket)

Private Member Functions

- void run ()

Private Attributes

- Socket * ReceiverSocket

- RTCPAbstractServer * Server
- double AverageRTCPSize

6.91.1 Detailed Description

RTCP Receiver.

This class implements an RTCP receiver based on Thread.

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

6.91.2 Constructor & Destructor Documentation

6.91.2.1 RTCPReceiver::RTCPReceiver ( )

Default constructor. You have to initialize RTCPReceiver by calling init(...) later!

See also

init

6.91.2.2 RTCPReceiver::RTCPReceiver (RTCPAbstractServer * *server*, Socket * *receiverSocket*)

Constructor for new RTCPReceiver. The new receiver's thread has to be started by calling start()!

Parameters

server	RTCPAbstractServer.
receiver-Socket	Socket to receive RTCP packets from.

6.91.2.3 RTCPReceiver::~RTCPReceiver ( )

Destructor.

6.91.3 Member Function Documentation

6.91.3.1 void RTCPReceiver::init (RTCPAbstractServer * *server*, Socket * *receiverSocket*)

Initialize new RTCPReceiver. The new receiver's thread has to be started by calling start()!

Parameters

server	RTCPAbstractServer.
receiverSocket	Socket to receive RTCP packets from.

6.91.3.2 void RTCPReceiver::run ( ) [private, virtual]

The virtual run() method, which contains the thread's implementation. It has to be implemented by classes, which inherit Thread.

Implements Thread.

6.91.4 Member Data Documentation

6.91.4.1 double RTCPReceiver::AverageRTCPSize [private]

6.91.4.2 Socket* RTCPReceiver::ReceiverSocket [private]

6.91.4.3 RTCPAbstractServer* RTCPReceiver::Server [private]

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

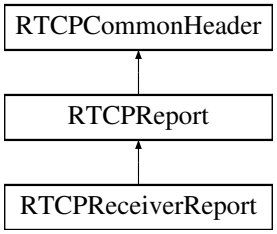
- rtcpreceiver.h
- rtcpreceiver.cc

6.92 RTCPReceiverReport Struct Reference

RTCP Sender Report.

`#include <rtcppacket.h>`

Inheritance diagram for RTCPReceiverReport:



Public Member Functions

- RTCPReceiverReport ()
- RTCPReceiverReport (const card32 syncSource, const card8 count=0)
- void init (const card32 syncSource, const card8 count=0)

Public Attributes

- RTCPReceptionReportBlock rr []

6.92.1 Detailed Description

RTCP Sender Report.

This struct manages an RTCP receiver report

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

See also

RTCPSender
RTCPReceiver
RTCPAbstractServer

6.92.2 Constructor & Destructor Documentation

6.92.2.1 RTCPReceiverReport::RTCPReceiverReport ( )

Constructor.

6.92.2.2 RTCPReceiverReport::RTCPReceiverReport (const card32 *syncSource*, const card8 *count* = 0)

Constructor.

Parameters

syncSource	SSRC.
count	Count.

6.92.3 Member Function Documentation

6.92.3.1 void RTCPReceiverReport::init (const card32 *syncSource*, const card8 *count* = 0)

Initialize.

Parameters

syncSource	SSRC.
count	Count.

6.92.4 Member Data Documentation

6.92.4.1 RTCPReceptionReportBlock RTCPReceiverReport::rr[]

Array of RTCPReceptionReportBlocks

The documentation for this struct was generated from the following files:

- rtcppacket.h
- rtcppacket.cc

6.93 RTCPReceptionReportBlock Struct Reference

RTCP Reception Report Block.

`#include <rtcppacket.h>`

Public Member Functions

- RTCPReceptionReportBlock ()
- RTCPReceptionReportBlock (const card32 ssrc)
- void init (const card32 ssrc)
- card32 getSSRC () const
- double getFractionLost () const
- card32 getPacketsLost () const
- card32 getLastSeqNum () const
- card32 getJitter () const
- card32 getLSR () const
- card32 getDLSR () const
- void setSSRC (card32 ssrc)
- void setFractionLost (const double fraction)
- void setPacketsLost (const card32 packetsLost)
- void setLastSeqNum (const card32 lastSeq)
- void setJitter (const card32 jitter)
- void setLSR (const card32 lsr)
- void setDLSR (const card32 dlsr)

Protected Attributes

- card32 SSRC
- card32 Fraction:8
- card32 Lost:24
- card32 LastSeq
- card32 Jitter
- card32 LSR
- card32 DLSR

6.93.1 Detailed Description

RTCP Reception Report Block.

This struct manages a reception report block

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

See also

RTCPSender
RTCPReceiver
RTCPAbstractServer

6.93.2 Constructor & Destructor Documentation

6.93.2.1 RTCPReceptionReportBlock::RTCPReceptionReportBlock ( )

Constructor.

6.93.2.2 RTCPReceptionReportBlock::RTCPReceptionReportBlock (const card32 *ssrc*)

Constructor.

Parameters

ssrc	SSRC.

6.93.3 Member Function Documentation

6.93.3.1 card32 RTCPReceptionReportBlock::getDLSR ( ) const [inline]

Get DLSR.

Returns

DLSR.

6.93.3.2 double RTCPReceptionReportBlock::getFractionLost ( ) const [inline]

Get fraction lost.

Returns

Fraction lost.

6.93.3.3 card32 RTCPReceptionReportBlock::getJitter ( ) const [inline]

Get jitter.

Returns

Jitter.

6.93.3.4 card32 RTCPReceptionReportBlock::getLastSeqNum ( ) const [inline]

Get last sequence number.

Returns

Last sequence number.

6.93.3.5 card32 RTCPReceptionReportBlock::getLSR ( ) const [inline]

Get LSR.

Returns

LSR.

6.93.3.6 card32 RTCPReceptionReportBlock::getPacketsLost ( ) const [inline]

Get packets lost.

Returns

Packets lost.

6.93.3.7 card32 RTCPReceptionReportBlock::getSSRC ( ) const [inline]

Get SSRC.

Returns

SSRC.

6.93.3.8 void RTCPReceptionReportBlock::init (const card32 *ssrc*)

Initialize.

Parameters

ssrc	SSRC.

6.93.3.9 void RTCPReceptionReportBlock::setDLSR (const card32 *dlsr*) [inline]

Set DLSR.

Parameters

dlsr	DLSR.

6.93.3.10 void RTCPReceptionReportBlock::setFractionLost (const double *fraction*) [inline]

Set fraction lost.

Parameters

fraction	Fraction lost.

6.93.3.11 void RTCPReceptionReportBlock::setJitter (const card32 *jitter*) [inline]

Set jitter.

Returns

jitter Jitter.

6.93.3.12 void RTCPReceptionReportBlock::setLastSeqNum (const card32 *lastSeq*) [inline]

Set last sequence number.

Parameters

lastSeq	Last sequence number.

6.93.3.13 void RTCPReceptionReportBlock::setLSR (const card32 *lsr*) [inline]

Set LSR.

Parameters

lsr	LSR.

6.93.3.14 void RTCPReceptionReportBlock::setPacketsLost (const card32 *packetsLost*) [inline]

Set packets lost.

Parameters

packetsLost	Packets lost.

6.93.3.15 void RTCPReceptionReportBlock::setSSRC (card32 *ssrc*) [inline]

Set SSRC.

Parameters

ssrc	SSRC.

6.93.4 Member Data Documentation

6.93.4.1 card32 RTCPReceptionReportBlock::DLSR [protected]

6.93.4.2 card32 RTCPReceptionReportBlock::Fraction [protected]

6.93.4.3 card32 RTCPReceptionReportBlock::Jitter [protected]

6.93.4.4 card32 RTCPReceptionReportBlock::LastSeq [protected]

6.93.4.5 card32 RTCPReceptionReportBlock::Lost [protected]

6.93.4.6 card32 RTCPReceptionReportBlock::LSR [protected]

6.93.4.7 card32 RTCPReceptionReportBlock::SSRC [protected]

The documentation for this struct was generated from the following files:

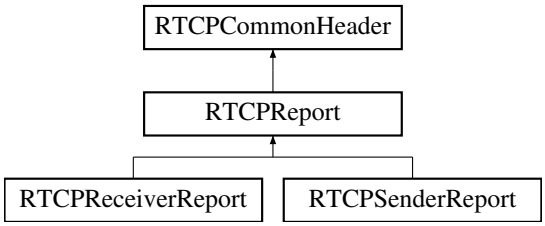
- rtcppacket.h
- rtcppacket.cc

6.94 RTCPReport Struct Reference

RTCP Report.

`#include <rtcppacket.h>`

Inheritance diagram for RTCPReport:



Public Member Functions

- RTCPReport ()
- card32 getSSRC () const
- void setSSRC (const card32 ssrc)

Protected Attributes

- card32 SSRC

6.94.1 Detailed Description

RTCP Report.

This struct manages an RTCP report.

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

See also

RTCPSender
RTCPReceiver
RTCPAbstractServer

6.94.2 Constructor & Destructor Documentation

6.94.2.1 RTCPReport::RTCPReport ( )

Constructor.

6.94.3 Member Function Documentation

6.94.3.1 card32 RTCPReport::getSSRC ( ) const [inline]

Get SSRC.

Returns

SSRC.

6.94.3.2 void RTCPReport::setSSRC (const card32 *ssrc*) [inline]

Set SSRC.

Parameters

ssrc	SSRC.

6.94.4 Member Data Documentation

6.94.4.1 card32 RTCPReport::SSRC [protected]

The documentation for this struct was generated from the following files:

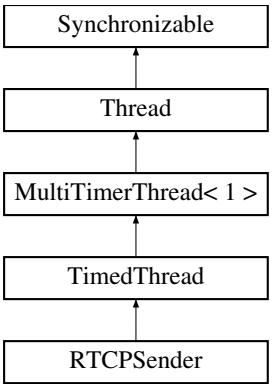
- rtcppacket.h
- rtcppacket.cc

6.95 RTCPSender Class Reference

RTCP Sender.

`#include <rtcpsender.h>`

Inheritance diagram for RTCPSender:



Public Member Functions

- RTCPSender ()
- RTCPSender (const InternetFlow &flow, const card32 ssrc, Socket *senderSocket, RTPReceiver *receiver, const card64 bandwidth, const card32 controlPPID)
- ~RTCPSender ()
- void init (const InternetFlow &flow, const card32 ssrc, Socket *senderSocket, RTPReceiver *receiver, const card64 bandwidth, const card32 controlPPID)
- integer sendApp (const char *name, const void *data, const cardinal dataLength)
- integer sendBye ()
- integer sendReport ()
- integer sendSDES ()
- bool addSDESItem (const card8 type, const void *data, const card8 length=0)
- void removeSDESItem (const card8 type)

Private Member Functions

- void timerEvent ()
- double computeTransmissionInterval ()

Private Attributes

- InternetFlow Flow
- SocketAddress * ReceiverAddress
- Socket * SenderSocket
- RTPReceiver * Receiver
- card32 SSRC
- std::multimap< const card8, RTCPSourceDescriptionItem * > SDESItemSet
- Randomizer Random
- card32 ControlPPID
- bool Initial
- bool WeSent
- integer Senders
- integer Members
- double RTCPBandwidth
- double AverageRTCPSize

6.95.1 Detailed Description

RTCP Sender.

This class implements an RTCP sender based on TimedThread.

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

6.95.2 Constructor & Destructor Documentation

6.95.2.1 RTCPSender::RTCPSender ( )

Default constructor. You have to initialize RTPSender by calling init(...) later!

See also

init

6.95.2.2 RTCPSender::RTCPSender (const InternetFlow & *flow*, const card32 *ssrc*, Socket * *senderSocket*, RTPReceiver * *receiver*, const card64 *bandwidth*, const card32 *controlPPID*)

Constructor for new RTCPSender. The new sender's thread has to be started by calling start()!

Parameters

flow	Flow to remote side.
ssrc	SSRC.
senderSocket	Socket to write data to.
receiver	RTPReceiver for reports to send.
bandwidth	RTCP Bandwidth (see RFC 1889).
controlPPID	PPID for SCTP transport.

6.95.2.3 RTCPSender::~RTCPSender ( )

Destructor.

6.95.3 Member Function Documentation

6.95.3.1 bool RTCPSender::addSDESItem (const card8 *type*, const void * *data*, const card8 *length* = 0)

Add SDES item to SDES item list. If a SDES item with the same type already exists in the list, the new item replaces the old item.

Parameters

type	SDES item type.
data	SDES item data.
length	SDES item data length.

Returns

true, if item has been added; false, if not.

See also

sendSDES

6.95.3.2 double RTCPSender::computeTransmissionInterval ( ) [private]

6.95.3.3 void RTCPSender::init (const InternetFlow & *flow,* const card32 *ssrc,* Socket * *senderSocket,* RTPReceiver * *receiver,* const card64 *bandwidth,* const card32 *controlPPID*)

Initialize new RTCPSender. The new sender's thread has to be started by calling start()!

Parameters

flow	Flow to remote side.
ssrc	SSRC.
senderSocket	Socket to write data to.
receiver	RTPReceiver for reports to send.
bandwidth	RTCP Bandwidth (see RFC 1889).
controlPPID	PPID for SCTP transport.

6.95.3.4 void RTCPSender::removeSDESItem (const card8 *type*)

Remove SDES item from SDES item list.

Parameters

type	SDES item type to be removed.

See also

addSDESItem
sendSDES

6.95.3.5 integer RTCPSender::sendApp (const char * *name,* const void * *data,* const cardinal *dataLength*)

Send RTCP APP message.

Parameters

name	RTCP APP name.
data	RTCP APP data.
dataLength	RTCP APP data length.

Returns

Bytes sent.

6.95.3.6 integer RTCPSender::sendBye ( )

Send RTCP BYE message.

Returns

Bytes sent.

6.95.3.7 integer RTCPSender::sendReport ( )

Send RTCP receiver report from the SourceStateInfo given in the constructor.

Returns

Bytes sent.

6.95.3.8 integer RTCPSender::sendSDES ( )

Send RTCP SDES message from the list given by addSDESItem().

Returns

Bytes sent.

See also

addSDESItem

6.95.3.9 void RTCPSender::timerEvent ( ) [private, virtual]

The virtual timerEvent() method, which contains the timed thread's implementation. It has to be implemented by classes, which inherit TimedThread. This method is called regularly with the given interval.

Implements TimedThread.

6.95.4 Member Data Documentation

6.95.4.1 double RTCPSender::AverageRTCPSize [private]

6.95.4.2 card32 RTCPSender::ControlPPID [private]

6.95.4.3 InternetFlow RTCPSender::Flow [private]

6.95.4.4 bool RTCPSender::Initial [private]

6.95.4.5 integer RTCPSender::Members [private]

6.95.4.6 Randomizer RTCPSender::Random [private]

6.95.4.7 RTPReceiver* RTCPSender::Receiver [private]

6.95.4.8 SocketAddress* RTCPSender::ReceiverAddress [private]

6.95.4.9 double RTCPSender::RTCPBandwidth [private]

6.95.4.10 std::multimap<const card8,RTCPSourceDescriptionItem*> RTCPSender::SDESItemSet [private]

6.95.4.11 integer RTCPSender::Senders [private]

6.95.4.12 Socket* RTCPSender::SenderSocket [private]

6.95.4.13 card32 RTCPSender::SSRC [private]

6.95.4.14 bool RTCPSender::WeSent [private]

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

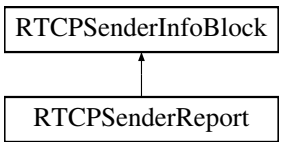
- rtcpsender.h
- rtcpsender.cc

6.96 RTCPSenderInfoBlock Struct Reference

RTCP Sender Info Block.

`#include <rtcppacket.h>`

Inheritance diagram for RTCPSenderInfoBlock:



Public Member Functions

- RTCPSenderInfoBlock ()
- card64 getNTPTimeStamp () const
- card32 getRTPTimeStamp () const
- card32 getPacketsSent () const
- card32 getOctetsSent () const
- void setNTPTimeStamp (const card64 timeStamp)
- void setRTPTimeStamp (const card32 timeStamp)
- void setPacketsSent (const card32 packets)
- void setOctetsSent (const card32 octets)

Protected Attributes

- card32 NTP_MostSignificant
- card32 NTP_LeastSignificant
- card32 RTPTimeStamp
- card32 PacketsSent
- card32 OctetsSent

6.96.1 Detailed Description

RTCP Sender Info Block.

This struct manages a sender info block

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

See also

RTCPSender
RTCPReceiver
RTCPAbstractServer

6.96.2 Constructor & Destructor Documentation

6.96.2.1 RTCPSenderInfoBlock::RTCPSenderInfoBlock ( )

Constructor.

6.96.3 Member Function Documentation

6.96.3.1 card64 RTCPSenderInfoBlock::getNTPTimeStamp ( ) const [inline]

Get NTP timestamp.

Returns

NTP timestamp.

6.96.3.2 card32 RTCPSenderInfoBlock::getOctetsSent ( ) const [inline]

Get octets sent.

Returns

Octets sent.

6.96.3.3 card32 RTCPSenderInfoBlock::getPacketsSent ( ) const [inline]

Get packets sent.

Returns

Packets sent.

6.96.3.4 card32 RTCPSenderInfoBlock::getRTPTimeStamp ( ) const [inline]

Get RTP time stamp.

Returns

RTP time stamp.

6.96.3.5 void RTCPSenderInfoBlock::setNTPTimeStamp (const card64 *timeStamp*) [inline]

Set NTP timestamp.

Parameters

timeStamp	NTP timestamp.

6.96.3.6 void RTCPSenderInfoBlock::setOctetsSent (const card32 *octets*) [inline]

Set octets sent.

Parameters

octets	Octets sent.

6.96.3.7 void RTCPSenderInfoBlock::setPacketsSent (const card32 *packets*) [inline]

Set packets sent.

Parameters

packets	Packets sent.

6.96.3.8 void RTCPSenderInfoBlock::setRTPTimeStamp (const card32 *timeStamp*) [inline]

Set RTP time stamp.

Parameters

timeStamp	RTP timestamp.

6.96.4 Member Data Documentation

6.96.4.1 card32 RTCPSenderInfoBlock::NTP_LeastSignificant [protected]

6.96.4.2 card32 RTCPSenderInfoBlock::NTP_MostSignificant [protected]

6.96.4.3 card32 RTCPSenderInfoBlock::OctetsSent [protected]

6.96.4.4 card32 RTCPSenderInfoBlock::PacketsSent [protected]

6.96.4.5 card32 RTCPSenderInfoBlock::RTPTimeStamp [protected]

The documentation for this struct was generated from the following files:

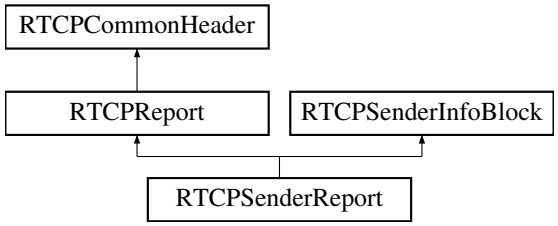
- rtcppacket.h
- rtcppacket.cc

6.97 RTCPSenderReport Struct Reference

RTCP Sender Report.

`#include <rtcppacket.h>`

Inheritance diagram for RTCPSenderReport:



Public Member Functions

- RTCPSenderReport ()
- RTCPSenderReport (const card32 syncSource, const card8 count=0)
- void init (const card32 syncSource, const card8 count=0)

Public Attributes

- RTCPReceptionReportBlock rr []

6.97.1 Detailed Description

RTCP Sender Report.

This struct manages an RTCP sender report

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

See also

RTCPSender
RTCPReceiver
RTCPAbstractServer

6.97.2 Constructor & Destructor Documentation

6.97.2.1 RTCPSenderReport::RTCPSenderReport ( )

Constructor.

6.97.2.2 RTCPSenderReport::RTCPSenderReport (const card32 *syncSource*, const card8 *count* = 0)

Constructor.

Parameters

syncSource	SSRC.
count	Count.

6.97.3 Member Function Documentation

6.97.3.1 void RTCPSenderReport::init (const card32 *syncSource*, const card8 *count* = 0)

Initialize.

Parameters

syncSource	SSRC.
count	Count.

6.97.4 Member Data Documentation

6.97.4.1 RTCPReceptionReportBlock RTCPSenderReport::rr[]

Array of RTCPReceptionReportBlocks

The documentation for this struct was generated from the following files:

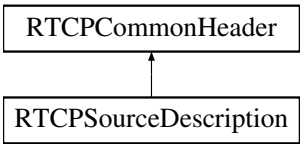
- rtcppacket.h
- rtcppacket.cc

6.98 RTCPSourceDescription Struct Reference

RTCP Source Description (SDES)

`#include <rtcppacket.h>`

Inheritance diagram for RTCPSourceDescription:



Public Member Functions

- RTCPSourceDescription ()
- RTCPSourceDescription (const card8 count)
- void init (const card8 count)

Public Attributes

- RTCPSourceDescriptionChunk Chunk [1]

6.98.1 Detailed Description

RTCP Source Description (SDES)

This struct manages an RTCP source description (SDES)

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

See also

RTCPSender
RTCPReceiver
RTCPAbstractServer

6.98.2 Constructor & Destructor Documentation

6.98.2.1 RTCPSourceDescription::RTCPSourceDescription ( )

Constructor.

6.98.2.2 RTCPSourceDescription::RTCPSourceDescription (const card8 *count*)

Constructor.

Parameters

Count	count.

6.98.3 Member Function Documentation

6.98.3.1 void RTCPSourceDescription::init (const card8 *count*)

Initialize.

Parameters

Count	count.

6.98.4 Member Data Documentation

6.98.4.1 RTCPSourceDescriptionChunk RTCPSourceDescription::Chunk[1]

Array of SDES chunks.

The documentation for this struct was generated from the following files:

- rtcppacket.h
- rtcppacket.cc

6.99 RTCPSourceDescriptionChunk Struct Reference

RTCP Source Description Chunk.

`#include <rtcppacket.h>`

Public Attributes

- card32 SRC
- RTCPSourceDescriptionItem Item []

6.99.1 Detailed Description

RTCP Source Description Chunk.

This struct manages an RTCP source description chunk

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

See also

RTCPSender
RTCPReceiver
RTCPAbstractServer

6.99.2 Member Data Documentation

6.99.2.1 RTCPSourceDescriptionItem RTCPSourceDescriptionChunk::Item[]

Array of SDES items.

6.99.2.2 card32 RTCPSourceDescriptionChunk::SRC

SSRC/CSRC Identifier of source.

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

- rtcppacket.h

6.100 RTCPSourceDescriptionItem Struct Reference

RTCP Source Description Item.

`#include <rtcppacket.h>`

Public Attributes

- card8 Type
- card8 Length
- char Data []

6.100.1 Detailed Description

RTCP Source Description Item.

This struct manages an RTCP source description item

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

See also

RTCPSender
RTCPReceiver
RTCPAbstractServer

6.100.2 Member Data Documentation

6.100.2.1 char RTCPSourceDescriptionItem::Data[]

Item data.

6.100.2.2 card8 RTCPSourceDescriptionItem::Length

Length in bytes.

6.100.2.3 card8 RTCPSourceDescriptionItem::Type

Item type (RTCP_SDES_...).

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

- rtcppacket.h

6.101 RTPAdaptionLayer Class Reference

Public Types

- enum DeleteReason { DeleteReason_UserBye = 0, DeleteReason_Timeout = 1, DeleteReason_Shutdown = 2, DeleteReason_Error = 3 }

Public Member Functions

- RTPAdaptionLayer (AbstractMediaServer *server)
- ~RTPAdaptionLayer ()
- void receivedSenderReport (const InternetFlow flow, const card32 source, const RTCPReceptionReportBlock *report, const cardinal layer)

- void receivedReceiverReport (const InternetFlow flow, const card32 source, const RTCPReceptionReportBlock *report, const cardinal layer)
- void receivedSourceDescription (const InternetFlow flow, const card32 source, const card8 type, const char *data, const card8 length)
- void receivedApp (const InternetFlow flow, const card32 source, const char *name, const void *data, const card32 dataLength)
- void receivedBye (const InternetFlow flow, const card32 source, const DeleteReason reason)
- String getIdentifier (const InternetFlow flow, const card32 source)

Private Attributes

- AbstractMediaServer * Server

6.101.1 Member Enumeration Documentation

6.101.1.1 enum RTPAdaptionLayer::DeleteReason

Enumerator:

DeleteReason_UserBye

DeleteReason_Timeout

DeleteReason_Shutdown

DeleteReason_Error

6.101.2 Constructor & Destructor Documentation

6.101.2.1 RTPAdaptionLayer::RTPAdaptionLayer (AbstractMediaServer * *server*)

6.101.2.2 RTPAdaptionLayer::~RTPAdaptionLayer ( )

6.101.3 Member Function Documentation

6.101.3.1 String RTPAdaptionLayer::getIdentifier (const InternetFlow *flow,* const card32 *source*)

6.101.3.2 void RTPAdaptionLayer::receivedApp (const InternetFlow *flow,* const card32 *source,* const char * *name,* const void * *data,* const card32 *dataLength*)

const discard!!!!

6.101.3.3 void RTPAdaptionLayer::receivedBye (const InternetFlow *flow,* const card32 *source,* const DeleteReason *reason*)

6.101.3.4 void RTPAdaptionLayer::receivedReceiverReport (const InternetFlow *flow,* const card32 *source,* const RTCPReceptionReportBlock * *report,* const cardinal *layer*)

6.101.3.5 void RTPAdaptionLayer::receivedSenderReport (const InternetFlow *flow,* const card32 *source,* const RTCPReceptionReportBlock * *report,* const cardinal *layer*)

6.101.3.6 void RTPAdaptionLayer::receivedSourceDescription (const InternetFlow *flow,* const card32 *source,* const card8 *type,* const char * *data,* const card8 *length*)

6.101.4 Member Data Documentation

6.101.4.1 AbstractMediaServer* RTPAdaptionLayer::Server [private]

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

- s2.cc

6.102 RTPPacket Struct Reference

RTP Packet.

`#include <rtppacket.h>`

Public Member Functions

- RTPPacket ()
- card8 getVersion () const
- card8 getPadding () const
- card8 getExtension () const
- card8 getCSRCCount () const
- bool getMarker () const
- card8 getPayloadType () const
- card16 getSequenceNumber () const
- card32 getTimeStamp () const
- card32 getSSRC () const
- card32 getCSRC (cardinal index) const
- cardinal calculateHeaderSize () const
- char * getPayloadData () const
- cardinal getMaxPayloadSize () const
- void setVersion (const card8 version)
- void setPadding (const card8 padding)
- void setExtension (const card8 extension)
- void setCSRCCount (const card8 count)

- void setMarker (const bool marker)
- void setPayloadType (const card8 payloadType)
- void setSequenceNumber (const card16 sequenceNumber)
- void setTimeStamp (const card32 timeStamp)
- void setSSRC (const card32 ssrc)
- void setCSRC (const cardinal index, const card32 csrc)

Private Attributes

- card8 V:2
- card8 P:1
- card8 X:1
- card8 CC:4
- card8 M:1
- card8 PT:7
- card16 SequenceNumber
- card32 TimeStamp
- card32 SSRC
- card32 CSRC [16]
- char Data [RTPConstants::RTPMaxPayloadLimit]

Friends

- std::ostream & operator<< (std::ostream &os, const RTPPacket &packet)

6.102.1 Detailed Description

RTP Packet.

This struct manages an RTP packet

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

See also

RTPSender
RTPReceiver

6.102.2 Constructor & Destructor Documentation

6.102.2.1 RTPPacket::RTPPacket ( )

Constructor.

6.102.3 Member Function Documentation

6.102.3.1 cardinal RTPPacket::calculateHeaderSize ( ) const [inline]

Calculate header size.

Returns

Header size.

6.102.3.2 card32 RTPPacket::getCSRC (cardinal *index*) const [inline]

Get CSRC at given index.

Parameters

index	Index.

Returns

RTP CSRC.

6.102.3.3 card8 RTPPacket::getCSRCCount ( ) const [inline]

Get CSRC count.

Returns

RTP CSRC count.

6.102.3.4 card8 RTPPacket::getExtension ( ) const [inline]

Get extension.

Returns

RTP Extension.

6.102.3.5 bool RTPPacket::getMarker ( ) const [inline]

Get marker.

Returns

RTP Marker.

6.102.3.6 cardinal RTPPacket::getMaxPayloadSize ( ) const [inline]

Get maximum payload size.

Returns

Maximum payload size.

6.102.3.7 card8 RTPPacket::getPadding ( ) const [inline]

Get padding.

Returns

RTP Padding.

6.102.3.8 char* RTPPacket::getPayloadData ( ) const [inline]

Get pointer to payload data.

Returns

pointer to payload data.

6.102.3.9 card8 RTPPacket::getPayloadType ( ) const [inline]

Get payload type.

Returns

RTP Payload type.

6.102.3.10 card16 RTPPacket::getSequenceNumber ( ) const [inline]

Get sequence number.

Returns

RTP Sequence number.

6.102.3.11 card32 RTPPacket::getSSRC ( ) const [inline]

Get SSRC.

Returns

RTP SSRC.

6.102.3.12 card32 RTPPacket::getTimeStamp ( ) const [inline]

Get time stamp.

Returns

RTP Time stamp.

6.102.3.13 card8 RTPPacket::getVersion ( ) const [inline]

Get version.

Returns

RTP Version.

6.102.3.14 void RTPPacket::setCSRC (const cardinal *index*, const card32 *csrc*) [inline]

Set CSRC at given index.

Parameters

index	Index.
csrc	CSRC.

6.102.3.15 void RTPPacket::setCSRCCount (const card8 *count*) [inline]

Set CSRC count.

Parameters

count	RTP CSRC count.

6.102.3.16 void RTPPacket::setExtension (const card8 *extension*) [inline]

Set extension.

Parameters

extension	RTP Extension.

6.102.3.17 void RTPPacket::setMarker (const bool *marker*) [inline]

Set marker.

Parameters

marker	RTP Marker.

6.102.3.18 void RTPPacket::setPadding (const card8 *padding*) [inline]

Set padding.

Parameters

padding	RTP Padding.

6.102.3.19 void RTPPacket::setPayloadType (const card8 *payloadType*) [inline]

Set payload type.

Parameters

payloadType	RTP Payload type.

6.102.3.20 void RTPPacket::setSequenceNumber (const card16 *sequenceNumber*) [inline]

Set sequence number.

Parameters

sequence-Number	RTP Sequence number.

6.102.3.21 void RTPPacket::setSSRC (const card32 *ssrc*) [inline]

Set SSRC.

Parameters

ssrc	RTP SSRC.

6.102.3.22 void RTPPacket::setTimeStamp (const card32 *timeStamp*) [inline]

Set time stamp.

Parameters

timeStamp	RTP timeStamp.

6.102.3.23 void RTPPacket::setVersion (const card8 *version*) [inline]

Set version.

Parameters

version	RTP Version.

6.102.4 Friends And Related Function Documentation

6.102.4.1 std::ostream& operator<< (std::ostream & *os*, const RTPPacket & *packet*) [friend]

Output operator.

6.102.5 Member Data Documentation

6.102.5.1 card8 RTPPacket::CC [private]

6.102.5.2 card32 RTPPacket::CSRC[16] [private]

6.102.5.3 char RTPPacket::Data[RTPConstants::RTPMaxPayloadLimit] [private]

6.102.5.4 card8 RTPPacket::M [private]

6.102.5.5 card8 RTPPacket::P [private]

6.102.5.6 card8 RTPPacket::PT [private]

6.102.5.7 card16 RTPPacket::SequenceNumber [private]

6.102.5.8 card32 RTPPacket::SSRC [private]

6.102.5.9 card32 RTPPacket::TimeStamp [private]

6.102.5.10 card8 RTPPacket::V [private]

6.102.5.11 card8 RTPPacket::X [private]

The documentation for this struct was generated from the following files:

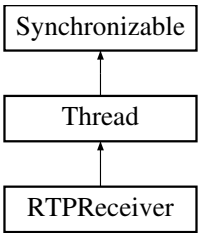
- rtppacket.h
- rtppacket.cc

6.103 RTPReceiver Class Reference

RTP Receiver.

`#include <rtpreceiver.h>`

Inheritance diagram for RTPReceiver:



Public Member Functions

- RTPReceiver ()
- RTPReceiver (DecoderInterface *decoder, Socket *receiverSocket)
- ~RTPReceiver ()
- void init (DecoderInterface *decoder, Socket *receiverSocket)
- card64 getPosition ()
- card64 getMaxPosition ()
- card64 getBytesReceived (const cardinal layer)
- card64 getPacketsReceived (const cardinal layer)
- void resetBytesReceived (const cardinal layer)
- void resetPacketsReceived (const cardinal layer)
- cardinal getLayers ()
- InternetFlow getInternetFlow (const cardinal layer=0)
- SourceStateInfo getSSI (const cardinal layer=0)

Protected Attributes

- cardinal Layers
- InternetFlow Flow [RTPConstants::RTPMaxQualityLayers]
- SourceStateInfo SSI [RTPConstants::RTPMaxQualityLayers]
- card64 BytesReceived [RTPConstants::RTPMaxQualityLayers]
- card64 PacketsReceived [RTPConstants::RTPMaxQualityLayers]

Private Member Functions

- void run ()

Private Attributes

- DecoderInterface * Decoder
- Socket * ReceiverSocket

Friends

- class RTCPSender

6.103.1 Detailed Description

RTP Receiver.

This class implements an RTP receiver based on Thread.

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

6.103.2 Constructor & Destructor Documentation

6.103.2.1 RTPReceiver::RTPReceiver ( )

Default constructor. You have to initialize RTPReceiver by calling init(...) later!

See also

init

6.103.2.2 RTPReceiver::RTPReceiver (DecoderInterface * *decoder,* Socket * *receiverSocket*)

Constructor for new RTPSender. The new sender's thread has to be started by calling start()!

Parameters

decoder	Decoder to handle packets received.
receiver-Socket	Socket to receive data from.

See also

Thread::start

6.103.2.3 RTPReceiver::~RTPReceiver ( )

Destructor.

6.103.3 Member Function Documentation

6.103.3.1 card64 RTPReceiver::getBytesReceived (const cardinal *layer*) [inline]

Get number of bytes received.

Parameters

layer	Layer number or (cardinal)-1 to get sum of all layers.

Returns

Bytes received.

6.103.3.2 InternetFlow RTPReceiver::getInternetFlow (const cardinal *layer* = 0) [inline]

Get InternetFlow of last transmission in a given layer.

Parameters

layer	Layer number.

Returns

InternetFlow.

6.103.3.3 cardinal RTPReceiver::getLayers ( ) [inline]

Get number of layers of last transmission.

Returns

Number of layers.

6.103.3.4 card64 RTPReceiver::getMaxPosition ( ) [inline]

Get maximum position of the encoder. The access is synchronized with the receiver thread.

Returns

Maximum position in nanoseconds.

6.103.3.5 card64 RTPReceiver::getPacketsReceived (const cardinal *layer*) [inline]

Get number of packets received.

Parameters

layer	Layer number or (cardinal)-1 to get sum of all layers.

Returns

Packets received.

6.103.3.6 card64 RTPReceiver::getPosition ( ) [inline]

Get position of the encoder. The access is synchronized with the receiver thread.

Returns

Position in nanoseconds.

6.103.3.7 SourceStateInfo RTPReceiver::getSSI (const cardinal *layer* = 0) [inline]

Get SourceStateInfo for given layer.

6.103.3.8 void RTPReceiver::init (DecoderInterface * *decoder*, Socket * *receiverSocket*)

Initialize RTPSender. The new receiver's thread has to be started by calling start()!

Parameters

decoder	Decoder to handle packets received.
receiver-Socket	Socket to receive data from.

See also

Thread::start

6.103.3.9 void RTPReceiver::resetBytesReceived (const cardinal *layer*) [inline]

Reset number of bytes received.

Parameters

layer	Layer number.

6.103.3.10 void RTPReceiver::resetPacketsReceived (const cardinal *layer*) [inline]

Reset number of packets received.

Parameters

layer	Layer number.

6.103.3.11 void RTPReceiver::run ( ) [private, virtual]

The virtual run() method, which contains the thread's implementation. It has to be implemented by classes, which inherit Thread.

Implements Thread.

6.103.4 Friends And Related Function Documentation

6.103.4.1 friend class RTCPSender [friend]

RTCPSender is a friend class to enable efficient update of SSI data.

6.103.5 Member Data Documentation

6.103.5.1 card64 RTPReceiver::BytesReceived[RTPConstants::RTPMaxQualityLayers] [protected]

6.103.5.2 DecoderInterface* RTPReceiver::Decoder [private]

6.103.5.3 InternetFlow RTPReceiver::Flow[RTPConstants::RTPMaxQualityLayers] [protected]

6.103.5.4 cardinal RTPReceiver::Layers [protected]

6.103.5.5 card64 RTPReceiver::PacketsReceived[RTPConstants::RTPMaxQualityLayers] [protected]

6.103.5.6 Socket* RTPReceiver::ReceiverSocket [private]

6.103.5.7 SourceStateInfo RTPReceiver::SSI[RTPConstants::RTPMaxQualityLayers] [protected]

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

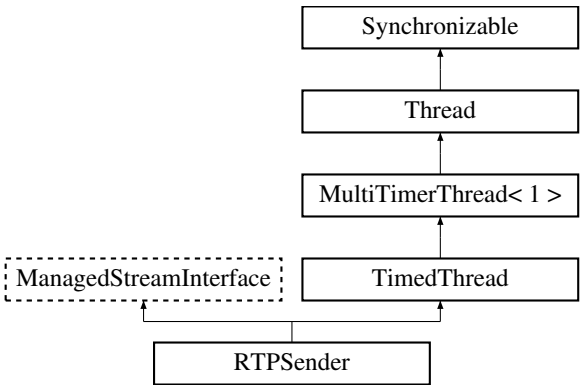
- rtpreceiver.h
- rtpreceiver.cc

6.104 RTPSender Class Reference

RTP Sender.

`#include <rtpsender.h>`

Inheritance diagram for RTPSender:



Public Member Functions

- RTPSender ()
- RTPSender (const InternetFlow &flow, const card32 ssrc, EncoderInterface ∗encoder, Socket ∗senderSocket, const card32 controlPPID, const card32 dataPPID, const cardinal maxPacketSize=1500, QoSManagerInterface ∗qosManager=NULL)
- ∼RTPSender ()
- void init (const InternetFlow &flow, const card32 ssrc, EncoderInterface ∗encoder, Socket ∗senderSocket, const card32 controlPPID, const card32 dataPPID, const cardinal maxPacketSize=1500, QoSManagerInterface ∗qosManager=NULL)
- AbstractQoSDescription ∗ getQoSDescription (const card64 offset)
- void updateQuality (const AbstractQoSDescription ∗aqd)
- void lock ()
- void unlock ()
- cardinal getMaxPacketSize () const
- cardinal setMaxPacketSize (const cardinal size)
- bool paused () const
- bool transmissionErrorDetected ()
- void setPause (const bool on)
- card64 getBytesSent ()
- card64 getPacketsSent ()
- void resetBytesSent ()
- void resetPacketsSent ()

Private Member Functions

- void timerEvent ()
- void updateFrameRate (const AbstractQoSDescription ∗aqd)

Private Attributes

- EncoderInterface ∗ Encoder
- Socket ∗ SenderSocket
- cardinal FramesPerSecond
- cardinal RenewCounter
- cardinal MaxPacketSize
- card32 SSRC
- card64 BytesSent
- card64 PacketsSent
- card64 TimeStamp
- card32 ControlPPID
- card32 DataPPID
- card32 PayloadBytesSent
- card32 PayloadPacketsSent
- bool Pause

- bool TransmissionError
- InternetFlow Flow [RTPConstants::RTPMaxQualityLayers]
- card16 SequenceNumber [RTPConstants::RTPMaxQualityLayers]
- QoSManagerInterface * QoSMgr
- cardinal Bandwidth [RTPConstants::RTPMaxQualityLayers]
- double BufferDelay [RTPConstants::RTPMaxQualityLayers]

6.104.1 Detailed Description

RTP Sender.

This class implements an RTP sender based on TimedThread.

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

6.104.2 Constructor & Destructor Documentation

6.104.2.1 RTPSender::RTPSender ( )

Default constructor. You have to initialize RTPSender by calling init(...) later!

See also

init

6.104.2.2 RTPSender::RTPSender (const InternetFlow & *flow*, const card32 *ssrc*, EncoderInterface * *encoder*, Socket * *senderSocket*, const card32 *controlPPID*, const card32 *dataPPID*, const cardinal *maxPacketSize* = 1500, QoSManagerInterface * *qosManager* = NULL)

Constructor for new RTPSender. The new sender's thread has to be started by calling start()!

Parameters

flow	Flow to remote side.
ssrc	Sender's SSRC (see RFC 1889).
encoder	Encoder to get packets to send from.
senderSocket	Socket to write packets to.
controlPPID	PPID for SCTP control transport.

dataPPID	PPID for SCTP data transport.
maxPacket-Size	Maximum packet size.
qosManager	QoS manager.

See also

Thread::start

6.104.2.3 RTPSender::~RTPSender ( )

Destructor.

6.104.3 Member Function Documentation

6.104.3.1 card64 RTPSender::getBytesSent ( ) [inline]

Get number of bytes sent.

Returns

Bytes sent.

6.104.3.2 cardinal RTPSender::getMaxPacketSize ( ) const [inline]

Get maximum packet size.

Returns

Maximum packet size.

6.104.3.3 card64 RTPSender::getPacketsSent ( ) [inline]

Get number of packets sent.

Returns

Packets sent.

6.104.3.4 AbstractQoSDescription * RTPSender::getQoSDescription (const card64 *offset*) [virtual]

Implementation of ManagedStreamInterface's getQoSDescription().

See also

ManagedStreamInterface::getQoSDescription

Implements ManagedStreamInterface.

6.104.3.5 void RTPSender::init (const InternetFlow & *flow*, const card32 *ssrc*, EncoderInterface * *encoder*, Socket * *senderSocket*, const card32 *controlPPID*, const card32 *dataPPID*, const cardinal *maxPacketSize* = 1500, QoSManagerInterface * *qosManager* = NULL)

Initialize new RTPSender. The new sender's thread has to be started by calling start()!

Parameters

flow	Flow to remote side.
ssrc	Sender's SSRC (see RFC 1889).
encoder	Encoder to get packets to send from.
senderSocket	Socket to write packets to.
controlPPID	PPID for SCTP control transport.
dataPPID	PPID for SCTP data transport.
maxPacketSize	Maximum packet size.
qosManager	QoS manager.

See also

Thread::start

6.104.3.6 void RTPSender::lock ( ) [virtual]

Implementation of ManagedStreamInterface's lock().

See also

ManagedStreamInterface::lock

Implements ManagedStreamInterface.

6.104.3.7 bool RTPSender::paused ( ) const [inline]

Check, if transmission is paused.

Returns

true, if paused; false otherwise.

6.104.3.8 void RTPSender::resetBytesSent ( ) [inline]

Reset number of bytes sent.

6.104.3.9 void RTPSender::resetPacketsSent ( ) [inline]

Reset number of packets sent.

6.104.3.10 cardinal RTPSender::setMaxPacketSize (const cardinal *size*) [inline]

Set maximum packet size.

Parameters

size	Maximum packet size.

Returns

Maximum packet size set.

6.104.3.11 void RTPSender::setPause (const bool *on*) [inline]

Set pause on or off.

Parameters

on	true to set pause on; false to set pause off.

6.104.3.12 void RTPSender::timerEvent ( ) [private, virtual]

The virtual timerEvent() method, which contains the timed thread's implementation. It has to be implemented by classes, which inherit TimedThread. This method is called regularly with the given interval.

Implements TimedThread.

6.104.3.13 bool RTPSender::transmissionErrorDetected ( ) [inline]

Check, if transmission error has been detected (e.g. destination rejects packets, no route etc.). Note: The transmission error attribute will be resetted to false by calling this

method.

6.104.3.14 void RTPSender::unlock ( ) [virtual]

Implementation of ManagedStreamInterface's unlock().

See also

ManagedStreamInterface::unlock

Implements ManagedStreamInterface.

6.104.3.15 void RTPSender::updateFrameRate (const AbstractQoSDescription * *aqd*) [private]

6.104.3.16 void RTPSender::updateQuality (const AbstractQoSDescription * *aqd*) [virtual]

Implementation of ManagedStreamInterface's updateQuality().

See also

ManagedStreamInterface::updateQuality

Implements ManagedStreamInterface.

6.104.4 Member Data Documentation

6.104.4.1 cardinal RTPSender::Bandwidth[RTPConstants::RTPMaxQualityLayers] [private]

6.104.4.2 double RTPSender::BufferDelay[RTPConstants::RTPMaxQualityLayers] [private]

6.104.4.3 card64 RTPSender::BytesSent [private]

6.104.4.4 card32 RTPSender::ControlPPID [private]

6.104.4.5 card32 RTPSender::DataPPID [private]

6.104.4.6 EncoderInterface* RTPSender::Encoder [private]

6.104.4.7 InternetFlow RTPSender::Flow[RTPConstants::RTPMaxQualityLayers] [private]

6.104.4.8 cardinal RTPSender::FramesPerSecond [private]

6.104.4.9 cardinal RTPSender::MaxPacketSize [private]

6.104.4.10 card64 RTPSender::PacketsSent [private]

6.104.4.11 bool RTPSender::Pause [private]

6.104.4.12 card32 RTPSender::PayloadBytesSent [private]

6.104.4.13 card32 RTPSender::PayloadPacketsSent [private]

6.104.4.14 QoSManagerInterface* RTPSender::QoSMgr [private]

6.104.4.15 cardinal RTPSender::RenewCounter [private]

6.104.4.16 Socket* RTPSender::SenderSocket [private]

6.104.4.17 card16 RTPSender::SequenceNumber[RTPConstants::RTPMaxQualityLayers] [private]

6.104.4.18 card32 RTPSender::SSRC [private]

6.104.4.19 card64 RTPSender::TimeStamp [private]

6.104.4.20 bool RTPSender::TransmissionError [private]

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

- rtpsender.h
- rtpsender.cc

6.105 sctp_adaptation_event Struct Reference

`#include <ext_socket.h>`

Public Attributes

- uint16_t sai_type
- uint16_t sai_flags
- uint32_t sai_length
- uint32_t sai_adaptation_ind
- sctp_assoc_t sai_assoc_id

6.105.1 Member Data Documentation

6.105.1.1 uint32_t sctp_adaptation_event::sai_adaptation_ind

6.105.1.2 sctp_assoc_t sctp_adaptation_event::sai_assoc_id

6.105.1.3 uint16_t sctp_adaptation_event::sai_flags

6.105.1.4 uint32_t sctp_adaptation_event::sai_length

6.105.1.5 uint16_t sctp_adaptation_event::sai_type

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

- ext_socket.h

6.106 sctp_assoc_change Struct Reference

`#include <ext_socket.h>`

Public Attributes

- uint16_t sac_type
- uint16_t sac_flags
- uint32_t sac_length
- uint16_t sac_state
- uint16_t sac_error
- uint16_t sac_outbound_streams
- uint16_t sac_inbound_streams
- sctp_assoc_t sac_assoc_id

6.106.1 Member Data Documentation

6.106.1.1 sctp_assoc_t sctp_assoc_change::sac_assoc_id

6.106.1.2 uint16_t sctp_assoc_change::sac_error

6.106.1.3 uint16_t sctp_assoc_change::sac_flags

6.106.1.4 uint16_t sctp_assoc_change::sac_inbound_streams

6.106.1.5 uint32_t sctp_assoc_change::sac_length

6.106.1.6 uint16_t sctp_assoc_change::sac_outbound_streams

6.106.1.7 uint16_t sctp_assoc_change::sac_state

6.106.1.8 uint16_t sctp_assoc_change::sac_type

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

- ext_socket.h

6.107 sctp_assoc_value Struct Reference

`#include <ext_socket.h>`

Public Attributes

- sctp_assoc_t assoc_id
- uint32_t assoc_value

6.107.1 Member Data Documentation

6.107.1.1 sctp_assoc_t sctp_assoc_value::assoc_id

6.107.1.2 uint32_t sctp_assoc_value::assoc_value

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

- ext_socket.h

6.108 sctp_assocparams Struct Reference

`#include <ext_socket.h>`

Public Attributes

- sctp_assoc_t sasoc_assoc_id
- uint16_t sasoc_asocmaxrxt
- uint16_t sasoc_number_peer_destinations
- uint32_t sasoc_peer_rwnd
- int32_t sasoc_local_rwnd
- uint32_t sasoc_cookie_life

6.108.1 Member Data Documentation

6.108.1.1 uint16_t sctp_assocparams::sasoc_asocmaxrxt

6.108.1.2 sctp_assoc_t sctp_assocparams::sasoc_assoc_id

6.108.1.3 uint32_t sctp_assocparams::sasoc_cookie_life

6.108.1.4 int32_t sctp_assocparams::sasoc_local_rwnd

6.108.1.5 uint16_t sctp_assocparams::sasoc_number_peer_destinations

6.108.1.6 uint32_t sctp_assocparams::sasoc_peer_rwnd

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

- ext_socket.h

6.109 sctp_data_arrive Struct Reference

```
#include <ext_socket.h>
```

Public Attributes

- uint16_t sda_type
- uint16_t sda_flags
- uint32_t sda_length
- sctp_assoc_t sda_assoc_id
- sctp_stream_t sda_stream
- uint32_t sda_ppid
- uint32_t sda_bytes_arrived

6.109.1 Member Data Documentation

6.109.1.1 sctp_assoc_t sctp_data_arrive::sda_assoc_id

6.109.1.2 uint32_t sctp_data_arrive::sda_bytes_arrived

6.109.1.3 uint16_t sctp_data_arrive::sda_flags

6.109.1.4 uint32_t sctp_data_arrive::sda_length

6.109.1.5 uint32_t sctp_data_arrive::sda_ppid

6.109.1.6 sctp_stream_t sctp_data_arrive::sda_stream

6.109.1.7 uint16_t sctp_data_arrive::sda_type

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

- ext_socket.h

6.110 sctp_event_subscribe Struct Reference

`#include <ext_socket.h>`

Public Attributes

- uint8_t sctp_data_io_event
- uint8_t sctp_association_event
- uint8_t sctp_address_event
- uint8_t sctp_send_failure_event
- uint8_t sctp_peer_error_event
- uint8_t sctp_shutdown_event
- uint8_t sctp_partial_delivery_event
- uint8_t sctp_adaptation_layer_event

6.110.1 Member Data Documentation

6.110.1.1 uint8_t sctp_event_subscribe::sctp_adaptation_layer_event

6.110.1.2 uint8_t sctp_event_subscribe::sctp_address_event

6.110.1.3 uint8_t sctp_event_subscribe::sctp_association_event

6.110.1.4 uint8_t sctp_event_subscribe::sctp_data_io_event

6.110.1.5 uint8_t sctp_event_subscribe::sctp_partial_delivery_event

6.110.1.6 uint8_t sctp_event_subscribe::sctp_peer_error_event

6.110.1.7 uint8_t sctp_event_subscribe::sctp_send_failure_event

6.110.1.8 uint8_t sctp_event_subscribe::sctp_shutdown_event

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

- ext_socket.h

6.111 sctp_initmsg Struct Reference

`#include <ext_socket.h>`

Public Attributes

- uint16_t sinit_num_ostreams
- uint16_t sinit_max_instreams
- uint16_t sinit_max_attempts
- uint16_t sinit_max_init_timeo

6.111.1 Member Data Documentation

6.111.1.1 uint16_t sctp_initmsg::sinit_max_attempts

6.111.1.2 uint16_t sctp_initmsg::sinit_max_init_timeo

6.111.1.3 uint16_t sctp_initmsg::sinit_max_instreams

6.111.1.4 uint16_t sctp_initmsg::sinit_num_ostreams

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

- ext_socket.h

6.112 sctp_notification Union Reference

`#include <ext_socket.h>`

Public Attributes

- struct {
  uint16_t sn_type
  uint16_t sn_flags
  uint32_t sn_length
  } sn_header

- struct sctp_assoc_change sn_assoc_change
- struct sctp_paddr_change sn_paddr_change
- struct sctp_remote_error sn_remote_error
- struct sctp_send_failed sn_send_failed
- struct sctp_shutdown_event sn_shutdown_event
- struct sctp_adaptation_event sn_adaptation_event
- struct sctp_pdapi_event sn_pdapi_event
- struct sctp_data_arrive sn_data_arrive

6.112.1 Member Data Documentation

6.112.1.1 struct sctp_adaptation_event sctp_notification::sn_adaptation_event

6.112.1.2 struct sctp_assoc_change sctp_notification::sn_assoc_change

6.112.1.3 struct sctp_data_arrive sctp_notification::sn_data_arrive

6.112.1.4 uint16_t sctp_notification::sn_flags

6.112.1.5 struct { ... } sctp_notification::sn_header

6.112.1.6 uint32_t sctp_notification::sn_length

6.112.1.7 struct sctp_paddr_change sctp_notification::sn_paddr_change

6.112.1.8 struct sctp_pdapi_event sctp_notification::sn_pdapi_event

6.112.1.9 struct sctp_remote_error sctp_notification::sn_remote_error

6.112.1.10 struct sctp_send_failed sctp_notification::sn_send_failed

6.112.1.11 struct sctp_shutdown_event sctp_notification::sn_shutdown_event

6.112.1.12 uint16_t sctp_notification::sn_type

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

- ext_socket.h

6.113 sctp_paddr_change Struct Reference

```
#include <ext_socket.h>
```

Public Attributes

- uint16_t spc_type
- uint16_t spc_flags
- uint32_t spc_length
- struct sockaddr_storage spc_aaddr
- int spc_state
- int spc_error
- sctp_assoc_t spc_assoc_id

6.113.1 Member Data Documentation

6.113.1.1 struct sockaddr_storage sctp_paddr_change::spc_aaddr

6.113.1.2 sctp_assoc_t sctp_paddr_change::spc_assoc_id

6.113.1.3 int sctp_paddr_change::spc_error

6.113.1.4 uint16_t sctp_paddr_change::spc_flags

6.113.1.5 uint32_t sctp_paddr_change::spc_length

6.113.1.6 int sctp_paddr_change::spc_state

6.113.1.7 uint16_t sctp_paddr_change::spc_type

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

- ext_socket.h

6.114 sctp_paddrinfo Struct Reference

```
#include <ext_socket.h>
```

Public Attributes

- sctp_assoc_t spinfo_assoc_id
- struct sockaddr_storage spinfo_address
- int32_t spinfo_state
- uint32_t spinfo_cwnd
- uint32_t spinfo_srtt
- uint32_t spinfo_rto
- uint32_t spinfo_mtu

6.114.1 Member Data Documentation

6.114.1.1 struct sockaddr_storage sctp_paddrinfo::spinfo_address

6.114.1.2 sctp_assoc_t sctp_paddrinfo::spinfo_assoc_id

6.114.1.3 uint32_t sctp_paddrinfo::spinfo_cwnd

6.114.1.4 uint32_t sctp_paddrinfo::spinfo_mtu

6.114.1.5 uint32_t sctp_paddrinfo::spinfo_rto

6.114.1.6 uint32_t sctp_paddrinfo::spinfo_srtt

6.114.1.7 int32_t sctp_paddrinfo::spinfo_state

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

- ext_socket.h

6.115 sctp_paddrparams Struct Reference

`#include <ext_socket.h>`

Public Attributes

- sctp_assoc_t spp_assoc_id
- struct sockaddr_storage spp_address
- uint32_t spp_hbinterval
- uint16_t spp_pathmaxrxt
- uint32_t spp_pathmtu
- uint32_t spp_sackdelay
- uint32_t spp_flags

6.115.1 Member Data Documentation

6.115.1.1 struct sockaddr_storage sctp_paddrparams::spp_address

6.115.1.2 sctp_assoc_t sctp_paddrparams::spp_assoc_id

6.115.1.3 uint32_t sctp_paddrparams::spp_flags

6.115.1.4 uint32_t sctp_paddrparams::spp_hbinterval

6.115.1.5 uint16_t sctp_paddrparams::spp_pathmaxrxt

6.115.1.6 uint32_t sctp_paddrparams::spp_pathmtu

6.115.1.7 uint32_t sctp_paddrparams::spp_sackdelay

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

- ext_socket.h

6.116 sctp_pdapi_event Struct Reference

`#include <ext_socket.h>`

Public Attributes

- uint16_t pdapi_type
- uint16_t pdapi_flags
- uint32_t pdapi_length
- uint32_t pdapi_indication
- sctp_assoc_t pdapi_assoc_id

6.116.1 Member Data Documentation

6.116.1.1 sctp_assoc_t sctp_pdapi_event::pdapi_assoc_id

6.116.1.2 uint16_t sctp_pdapi_event::pdapi_flags

6.116.1.3 uint32_t sctp_pdapi_event::pdapi_indication

6.116.1.4 uint32_t sctp_pdapi_event::pdapi_length

6.116.1.5 uint16_t sctp_pdapi_event::pdapi_type

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

- ext_socket.h

6.117 sctp_remote_error Struct Reference

`#include <ext_socket.h>`

Public Attributes

- uint16_t sre_type
- uint16_t sre_flags
- uint32_t sre_length
- uint16_t sre_error
- sctp_assoc_t sre_assoc_id
- uint8_t sre_data [0]

6.117.1 Member Data Documentation

6.117.1.1 sctp_assoc_t sctp_remote_error::sre_assoc_id

6.117.1.2 uint8_t sctp_remote_error::sre_data[0]

6.117.1.3 uint16_t sctp_remote_error::sre_error

6.117.1.4 uint16_t sctp_remote_error::sre_flags

6.117.1.5 uint32_t sctp_remote_error::sre_length

6.117.1.6 uint16_t sctp_remote_error::sre_type

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

- ext_socket.h

6.118 sctp_rtoinfo Struct Reference

`#include <ext_socket.h>`

Public Attributes

- sctp_assoc_t srto_assoc_id
- uint32_t srto_initial
- uint32_t srto_max
- uint32_t srto_min

6.118.1 Member Data Documentation

6.118.1.1 sctp_assoc_t sctp_rtoinfo::srto_assoc_id

6.118.1.2 uint32_t sctp_rtoinfo::srto_initial

6.118.1.3 uint32_t sctp_rtoinfo::srto_max

6.118.1.4 uint32_t sctp_rtoinfo::srto_min

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

- ext_socket.h

6.119 sctp_sack_info Struct Reference

`#include <ext_socket.h>`

Public Attributes

- sctp_assoc_t sack_assoc_id
- uint32_t sack_delay
- uint32_t sack_freq

6.119.1 Member Data Documentation

6.119.1.1 sctp_assoc_t sctp_sack_info::sack_assoc_id

6.119.1.2 uint32_t sctp_sack_info::sack_delay

6.119.1.3 uint32_t sctp_sack_info::sack_freq

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

- ext_socket.h

6.120 sctp_send_failed Struct Reference

`#include <ext_socket.h>`

Public Attributes

- uint16_t ssf_type
- uint16_t ssf_flags
- uint32_t ssf_length
- uint32_t ssf_error
- struct sctp_sndrcvinfo ssf_info
- sctp_assoc_t ssf_assoc_id
- uint8_t ssf_data [0]

6.120.1 Member Data Documentation

6.120.1.1 sctp_assoc_t sctp_send_failed::ssf_assoc_id

6.120.1.2 uint8_t sctp_send_failed::ssf_data[0]

6.120.1.3 uint32_t sctp_send_failed::ssf_error

6.120.1.4 uint16_t sctp_send_failed::ssf_flags

6.120.1.5 struct sctp_sndrcvinfo sctp_send_failed::ssf_info

6.120.1.6 uint32_t sctp_send_failed::ssf_length

6.120.1.7 uint16_t sctp_send_failed::ssf_type

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

- ext_socket.h

6.121 sctp_setpeerprim Struct Reference

`#include <ext_socket.h>`

Public Attributes

- sctp_assoc_t sspp_assoc_id
- struct sockaddr_storage sspp_addr

6.121.1 Member Data Documentation

6.121.1.1 struct sockaddr_storage sctp_setpeerprim::sspp_addr

6.121.1.2 sctp_assoc_t sctp_setpeerprim::sspp_assoc_id

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

- ext_socket.h

6.122 sctp_setprim Struct Reference

`#include <ext_socket.h>`

Public Attributes

- sctp_assoc_t ssp_assoc_id
- struct sockaddr_storage ssp_addr

6.122.1 Member Data Documentation

6.122.1.1 struct sockaddr_storage sctp_setprim::ssp_addr

6.122.1.2 sctp_assoc_t sctp_setprim::ssp_assoc_id

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

- ext_socket.h

6.123 sctp_setstrm_timeout Struct Reference

`#include <ext_socket.h>`

Public Attributes

- sctp_assoc_t ssto_assoc_id
- uint32_t ssto_timeout
- uint16_t ssto_streamid_start
- uint16_t ssto_streamid_end

6.123.1 Member Data Documentation

6.123.1.1 sctp_assoc_t sctp_setstrm_timeout::ssto_assoc_id

6.123.1.2 uint16_t sctp_setstrm_timeout::ssto_streamid_end

6.123.1.3 uint16_t sctp_setstrm_timeout::ssto_streamid_start

6.123.1.4 uint32_t sctp_setstrm_timeout::ssto_timeout

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

- ext_socket.h

6.124 sctp_shutdown_event Struct Reference

`#include <ext_socket.h>`

Public Attributes

- uint16_t sse_type
- uint16_t sse_flags

- uint32_t sse_length
- sctp_assoc_t sse_assoc_id

6.124.1 Member Data Documentation

6.124.1.1 sctp_assoc_t sctp_shutdown_event::sse_assoc_id

6.124.1.2 uint16_t sctp_shutdown_event::sse_flags

6.124.1.3 uint32_t sctp_shutdown_event::sse_length

6.124.1.4 uint16_t sctp_shutdown_event::sse_type

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

- ext_socket.h

6.125 sctp_sndrcvinfo Struct Reference

`#include <ext_socket.h>`

Public Attributes

- uint16_t sinfo_stream
- uint16_t sinfo_ssn
- uint32_t sinfo_flags
- uint32_t sinfo_ppid
- uint32_t sinfo_context
- uint32_t sinfo_timetolive
- uint32_t sinfo_tsn
- uint32_t sinfo_cumtsn
- sctp_assoc_t sinfo_assoc_id

6.125.1 Member Data Documentation

6.125.1.1 sctp_assoc_t sctp_sndrcvinfo::sinfo_assoc_id

6.125.1.2 uint32_t sctp_sndrcvinfo::sinfo_context

6.125.1.3 uint32_t sctp_sndrcvinfo::sinfo_cumtsn

6.125.1.4 uint32_t sctp_sndrcvinfo::sinfo_flags

6.125.1.5 uint32_t sctp_sndrcvinfo::sinfo_ppid

6.125.1.6 uint16_t sctp_sndrcvinfo::sinfo_ssn

6.125.1.7 uint16_t sctp_sndrcvinfo::sinfo_stream

6.125.1.8 uint32_t sctp_sndrcvinfo::sinfo_timetolive

6.125.1.9 uint32_t sctp_sndrcvinfo::sinfo_tsn

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

- ext_socket.h

6.126 sctp_status Struct Reference

```
#include <ext_socket.h>
```

Public Attributes

- sctp_assoc_t sstat_assoc_id
- int32_t sstat_state
- uint32_t sstat_rwnd
- uint16_t sstat_unackdata
- uint16_t sstat_penddata
- uint16_t sstat_instrms
- uint16_t sstat_outstrms
- uint32_t sstat_fragmentation_point
- struct sctp_paddrinfo sstat_primary

6.126.1 Member Data Documentation

6.126.1.1 sctp_assoc_t sctp_status::sstat_assoc_id

6.126.1.2 uint32_t sctp_status::sstat_fragmentation_point

6.126.1.3 uint16_t sctp_status::sstat_instrms

6.126.1.4 uint16_t sctp_status::sstat_outstrms

6.126.1.5 uint16_t sctp_status::sstat_penddata

6.126.1.6 struct sctp_paddrinfo sctp_status::sstat_primary

6.126.1.7 uint32_t sctp_status::sstat_rwnd

6.126.1.8 int32_t sctp_status::sstat_state

6.126.1.9 uint16_t sctp_status::sstat_unackdata

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

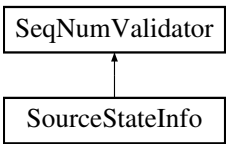
- ext_socket.h

6.127 SeqNumValidator Class Reference

Sequence Number Validator.

`#include <seqnumvalidator.h>`

Inheritance diagram for SeqNumValidator:



Public Types

- enum ValidationResult { Valid = 0, SourceProbation = 1, Jumped = 2, Invalid = 10, DuplicatePacket = Invalid + 0, InvalidSeqNum = Invalid + 1 }

Public Member Functions

- SeqNumValidator (const cardinal minSequential=2, const cardinal maxMisorder=100, const cardinal maxDropout=3000, const card64 seqMod=(1<< 16))
- card64 getPacketsReceived () const
- card64 getPacketsLost () const
- card64 getLastSeqNum () const
- double getFractionLost () const
- double getJitter () const
- ValidationResult validate (const card64 sequenceNumber, const card32 packetTimeStamp=0)
- void reset ()
- double calculateFractionLost ()

Private Member Functions

- void init (const card64 sequenceNumber)

Private Attributes

- card64 SeqMod
- cardinal MaxDropout
- cardinal MaxMisorder
- cardinal MinSequential
- card64 PrevPacketTimeStamp
- card64 PrevPacketArrivalTime
- double Jitter
- double FractionLost
- card64 MaxSeq
- card64 BaseSeq
- card64 BadSeq
- card32 Probation
- card64 Cycles
- card64 Received
- card64 ReceivedPrior
- card64 ExpectedPrior
- bool Uninitialized

6.127.1 Detailed Description

Sequence Number Validator.

This class is a validator for sequence numbers. It is based on the algorithm described in RFC 1889. It can use sequence numbers up to a size of 64 bits. Jitter and fraction loss calculation is also done by this class.

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

6.127.2 Member Enumeration Documentation

6.127.2.1 enum SeqNumValidator::ValidationResult

Enumerator:

Valid

SourceProbation

Jumped

Invalid

DuplicatePacket

InvalidSeqNum

6.127.3 Constructor & Destructor Documentation

6.127.3.1 SeqNumValidator::SeqNumValidator (const cardinal *minSequential* = 2, const cardinal *maxMisorder* = 100, const cardinal *maxDropout* = 3000, const card64 *seqMod* = (1 << 16))

Constructor for new sequence number validator.

Parameters

minSequential	Minimum number of packets in sequence for the source to be valid.
maxMisorder	Maximum difference for packets to be misordered.
maxDropout	Maximum gap.
seqMod	Sequence number modulo.

6.127.4 Member Function Documentation

6.127.4.1 double SeqNumValidator::calculateFractionLost ( )

Calculate and get fraction of packets lost.

Returns

Fraction lost.

6.127.4.2 double SeqNumValidator::getFractionLost ( ) const [inline]

Get fraction of packets lost. Note: No calculation of the fraction lost is done here! The fraction lost value is the value of the last call of calculateFractionLost()!

Returns

Fraction of packets lost.

See also

calculateFractionLost

6.127.4.3 double SeqNumValidator::getJitter ( ) const [inline]

Get jitter.

Returns

Jitter.

6.127.4.4 card64 SeqNumValidator::getLastSeqNum ( ) const [inline]

Get extended last sequence number. This number is extended by the calculated number of sequence number cycles!

Returns

Last sequence number.

6.127.4.5 card64 SeqNumValidator::getPacketsLost ( ) const [inline]

Get number of packets lost. The loss is calculated by the number of sequence number cycles and gaps.

Returns

Number of packets lost.

6.127.4.6 card64 SeqNumValidator::getPacketsReceived ( ) const [inline]

Get number of packets received.

Returns

Number of packets received.

6.127.4.7 void SeqNumValidator::init (const card64 *sequenceNumber*) [inline, private]

6.127.4.8 void SeqNumValidator::reset ( )

Reset SeqNumValidator.

Reimplemented in SourceStateInfo.

6.127.4.9 SeqNumValidator::ValidationResult SeqNumValidator::validate (const card64 *sequenceNumber,* const card32 *packetTimeStamp* = 0)

Validate a new sequence number. If the packet is valid, jitter value will be calculated using packetTimeStamp. To disable jitter calculation, set packetTimeStamp to 0.

Parameters

sequence-Number	Sequence number to be validated.
packetTime-Stamp	Time stamp of the packet for jitter calculation.

Returns

ValidationResult containing result of validation.

6.127.5 Member Data Documentation

6.127.5.1 card64 SeqNumValidator::BadSeq [private]

6.127.5.2 card64 SeqNumValidator::BaseSeq [private]

6.127.5.3 card64 SeqNumValidator::Cycles [private]

6.127.5.4 card64 SeqNumValidator::ExpectedPrior [private]

Reimplemented in SourceStateInfo.

6.127.5.5 double SeqNumValidator::FractionLost [private]

Reimplemented in SourceStateInfo.

6.127.5.6 double SeqNumValidator::Jitter [private]

6.127.5.7 cardinal SeqNumValidator::MaxDropout [private]

6.127.5.8 cardinal SeqNumValidator::MaxMisorder [private]

6.127.5.9 card64 SeqNumValidator::MaxSeq [private]

6.127.5.10 cardinal SeqNumValidator::MinSequential [private]

6.127.5.11 card64 SeqNumValidator::PrevPacketArrivalTime [private]

6.127.5.12 card64 SeqNumValidator::PrevPacketTimeStamp [private]

6.127.5.13 card32 SeqNumValidator::Probation [private]

6.127.5.14 card64 SeqNumValidator::Received [private]

6.127.5.15 card64 SeqNumValidator::ReceivedPrior [private]

Reimplemented in SourceStateInfo.

6.127.5.16 card64 SeqNumValidator::SeqMod [private]

6.127.5.17 bool SeqNumValidator::Uninitialized [private]

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

- seqnumvalidator.h
- seqnumvalidator.cc

6.128 ServiceLevelAgreement Class Reference

Trace Layer Configuration.

`#include <servicelevelagreement.h>`

Public Member Functions

- ServiceLevelAgreement ()
- ~ServiceLevelAgreement ()
- bool load (const char *fileName)
- cardinal getPossibleClassesForBandwidthInfo (const AbstractLayerDescription *ald, cardinal *classList) const

Public Attributes

- card64 TotalBandwidth
- cardinal BestEffort
- cardinal Classes
- DiffServClass Class [TrafficClassValues::MaxValues]

6.128.1 Detailed Description

Trace Layer Configuration.

This class is a service level agreement (SLA).

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

6.128.2 Constructor & Destructor Documentation

6.128.2.1 ServiceLevelAgreement::ServiceLevelAgreement ( )

Constructor.

6.128.2.2 ServiceLevelAgreement::~ServiceLevelAgreement ( )

Destructor.

6.128.3 Member Function Documentation

6.128.3.1 cardinal ServiceLevelAgreement::getPossibleClassesForBandwidthInfo (const AbstractLayerDescription * *ald*, cardinal * *classList*) const

Get possible DiffServ classes for given bandwidth info.

Parameters

ald	AbstractLayerDescription.
classList	Array to store class index numbers.

Returns

Number of class index numbers stored (0 = no possible classes found).

6.128.3.2 bool ServiceLevelAgreement::load (const char * *fileName*)

Load configuration from file.

6.128.4 Member Data Documentation

6.128.4.1 cardinal ServiceLevelAgreement::BestEffort

6.128.4.2 DiffServClass ServiceLevelAgreement::Class[TrafficClassValues::MaxValues]

6.128.4.3 cardinal ServiceLevelAgreement::Classes

6.128.4.4 card64 ServiceLevelAgreement::TotalBandwidth

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

- servicelevelagreement.h
- servicelevelagreement.cc

6.129 SessionDescription Struct Reference

Session Description.

`#include <sessiondescription.h>`

Public Attributes

- cardinal SessionID
- cardinal Streams
- std::multimap < ManagedStreamInterface *, StreamDescription * > StreamSet
- card64 MinWantedBandwidth
- card64 MaxWantedBandwidth
- card64 TotalAllocatedBandwidth
- card64 AllocatedBandwidthArray [TrafficClassValues::MaxValues]
- int8 Priority
- bool MaximumReached

6.129.1 Detailed Description

Session Description.

This structure contains a description of a session. It is used for the bandwidth manager's remapping algorithm.

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

6.129.2 Member Data Documentation

6.129.2.1 card64 SessionDescription::AllocatedBandwidthArray[TrafficClassValues::MaxValues]

Allocated bandwidth for each class.

6.129.2.2 bool SessionDescription::MaximumReached

True, if all following higher bandwidth allocations will fail (no more bandwidth available to achieve higher quality -> no more allocation trials necessary); false otherwise.

6.129.2.3 card64 SessionDescription::MaxWantedBandwidth

Maximum session bandwidth.

6.129.2.4 card64 SessionDescription::MinWantedBandwidth

Minimum session bandwidth.

6.129.2.5 int8 SessionDescription::Priority

Session priority.

6.129.2.6 cardinal SessionDescription::SessionID

Session ID.

6.129.2.7 cardinal SessionDescription::Streams

Number of streams.

6.129.2.8 std::multimap<ManagedStreamInterface*,StreamDescription*> SessionDescription::StreamSet

Stream description set.

6.129.2.9 card64 SessionDescription::TotalAllocatedBandwidth

Total allocated bandwidth.

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

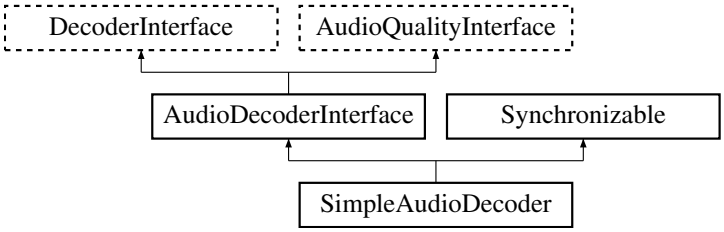
- sessiondescription.h

6.130 SimpleAudioDecoder Class Reference

Simple Audio Decoder.

`#include <simpleaudiodecoder.h>`

Inheritance diagram for SimpleAudioDecoder:



Public Member Functions

- SimpleAudioDecoder (AudioWriterInterface *audioWriter)
- ~SimpleAudioDecoder ()
- const card16 getTypeID () const
- const char * getTypeName () const
- void activate ()
- void deactivate ()
- void reset ()
- void getMediaInfo (MediaInfo &mediaInfo) const
- card8 getErrorCode () const
- card64 getPosition () const
- card64 getMaxPosition () const
- bool checkNextPacket (DecoderPacket *decoderPacket)
- void handleNextPacket (const DecoderPacket *decoderPacket)
- card16 getSamplingRate () const
- card8 getBits () const
- card8 getChannels () const
- card16 getByteOrder () const
- cardinal getBytesPerSecond () const
- cardinal getBitsPerSample () const
- AudioQuality getWantedQuality () const
- void setWantedQuality (const AudioQualityInterface &wantedQuality)

Private Attributes

- SeqNumValidator SeqNumber
- AudioWriterInterface * Device
- card64 Position
- card64 MaxPosition
- AudioQuality WantedQuality
- MediaInfo Media
- card16 AudioSamplingRate
- card8 AudioBits
- card8 AudioChannels
- card8 ErrorCode

6.130.1 Detailed Description

Simple Audio Decoder.

This class is an simple audio decoder. It does no error correction or redundant transmission.

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

6.130.2 Constructor & Destructor Documentation

6.130.2.1 SimpleAudioDecoder::SimpleAudioDecoder (AudioWriterInterface * *audioWriter*)

Constructor for the audio decoder.

Parameters

audioWriter	AudioReaderInterface for the audio output.

6.130.2.2 SimpleAudioDecoder::~SimpleAudioDecoder ( )

Destructor.

6.130.3 Member Function Documentation

6.130.3.1 void SimpleAudioDecoder::activate ( ) [virtual]

activate() implementation of DecoderInterface.

See also

DecoderInterface::activate

Implements DecoderInterface.

6.130.3.2 bool SimpleAudioDecoder::checkNextPacket (DecoderPacket * *decoderPacket*) [virtual]

checkNextPacket() implementation of DecoderInterface.

See also

DecoderInterface::checkNextPacket

Implements DecoderInterface.

6.130.3.3 void SimpleAudioDecoder::deactivate ( ) [virtual]

deactivate() implementation of DecoderInterface.

See also

DecoderInterface::deactivate

Implements DecoderInterface.

6.130.3.4 card8 SimpleAudioDecoder::getBits ( ) const [virtual]

getBits() Implementation of AudioDecoderInterface.

See also

AudioDecoderInterface::getBits

Implements AudioQualityInterface.

6.130.3.5 cardinal SimpleAudioDecoder::getBitsPerSample ( ) const [virtual]

getBitsPerSample() implementation of AudioDecoderInterface.

See also

AudioDecoderInterface::getBitsPerSample

Implements AudioQualityInterface.

6.130.3.6 card16 SimpleAudioDecoder::getByteOrder ( ) const [virtual]

getByteOrder() Implementation of AudioQualityInterface.

See also

AudioQualityInterface::getByteOrder

Implements AudioQualityInterface.

6.130.3.7 cardinal SimpleAudioDecoder::getBytesPerSecond ( ) const [virtual]

getBytesPerSecond() implementation of AudioDecoderInterface.

See also

AudioDecoderInterface::getBytesPerSecond

Implements AudioQualityInterface.

6.130.3.8 card8 SimpleAudioDecoder::getChannels ( ) const [virtual]

getChannels() Implementation of AudioDecoderInterface.

See also

AudioDecoderInterface::getChannels

Implements AudioQualityInterface.

6.130.3.9 card8 SimpleAudioDecoder::getErrorCode ( ) const [virtual]

getErrorCode() implementation of DecoderInterface.

See also

DecoderInterface::getErrorCode

Implements DecoderInterface.

6.130.3.10 card64 SimpleAudioDecoder::getMaxPosition ( ) const [virtual]

getMaxPosition() implementation of DecoderInterface.

See also

DecoderInterface::getMaxPosition

Implements DecoderInterface.

6.130.3.11 void SimpleAudioDecoder::getMediaInfo (MediaInfo & *mediaInfo*) const [virtual]

getMediaInfo() implementation of DecoderInterface.

See also

DecoderInterface::getMediaInfo

Implements DecoderInterface.

6.130.3.12 card64 SimpleAudioDecoder::getPosition ( ) const [virtual]

getPosition() implementation of DecoderInterface.

See also

DecoderInterface::getPosition

Implements DecoderInterface.

6.130.3.13 card16 SimpleAudioDecoder::getSamplingRate ( ) const [virtual]

getSamplingRate() Implementation of AudioDecoderInterface.

See also

AudioDecoderInterface::getSamplingRate

Implements AudioQualityInterface.

6.130.3.14 const card16 SimpleAudioDecoder::getTypeID ( ) const [virtual]

getTypeID() implementation of DecoderInterface.

See also

DecoderInterface::getTypeID

Implements DecoderInterface.

6.130.3.15 const char * SimpleAudioDecoder::getTypeName ( ) const [virtual]

getTypeName implementation of DecoderInterface.

See also

DecoderInterface::getTypeName

Implements DecoderInterface.

6.130.3.16 AudioQuality SimpleAudioDecoder::getWantedQuality ( ) const [virtual]

getWantedQuality() implementation of AudioDecoderInterface.

see AudioDecoderInterface::getWantedQuality

Implements AudioDecoderInterface.

6.130.3.17 void SimpleAudioDecoder::handleNextPacket (const DecoderPacket * *decoderPacket*) [virtual]

handleNextPacket() implementation of DecoderInterface.

See also

DecoderInterface::handleNextPacket

Implements DecoderInterface.

6.130.3.18 void SimpleAudioDecoder::reset ( ) [virtual]

reset() implementation of DecoderInterface.

See also

DecoderInterface::reset

Implements DecoderInterface.

6.130.3.19 void SimpleAudioDecoder::setWantedQuality (const AudioQualityInterface & *wantedQuality*) [virtual]

setWantedQuality() implementation of AudioDecoderInterface.

See also

AudioDecoderInterface::setWantedQuality

Implements AudioDecoderInterface.

6.130.4 Member Data Documentation

6.130.4.1 card8 SimpleAudioDecoder::AudioBits [private]

6.130.4.2 card8 SimpleAudioDecoder::AudioChannels [private]

6.130.4.3 card16 SimpleAudioDecoder::AudioSamplingRate [private]

6.130.4.4 AudioWriterInterface* SimpleAudioDecoder::Device [private]

6.130.4.5 card8 SimpleAudioDecoder::ErrorCode [private]

6.130.4.6 card64 SimpleAudioDecoder::MaxPosition [private]

6.130.4.7 MediaInfo SimpleAudioDecoder::Media [private]

6.130.4.8 card64 SimpleAudioDecoder::Position [private]

6.130.4.9 SeqNumValidator SimpleAudioDecoder::SeqNumber [private]

6.130.4.10 AudioQuality SimpleAudioDecoder::WantedQuality [private]

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

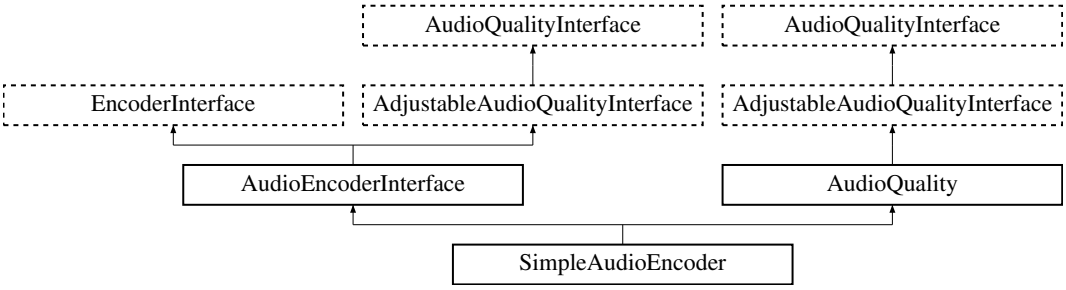
- simpleaudiodecoder.h
- simpleaudiodecoder.cc

6.131 SimpleAudioEncoder Class Reference

Simple Audio Encoder.

`#include <simpleaudioencoder.h>`

Inheritance diagram for SimpleAudioEncoder:



Public Member Functions

- SimpleAudioEncoder (AudioReaderInterface *audioReader)
- ~SimpleAudioEncoder ()
- const card16 getTypeID () const
- const char * getTypeName () const
- void activate ()
- void deactivate ()
- void reset ()
- bool checkInterval (card64 &time, bool &newRUList)
- bool prepareNextFrame (const cardinal headerSize, const cardinal maxPacket-Size, const cardinal flags)
- cardinal getNextPacket (EncoderPacket *encoderPacket)
- double getFrameRate () const
- AbstractQoSDescription * getQoSDescription (const cardinal pktHeaderSize, const cardinal pktMaxSize, const card64 offset)
- void updateQuality (const AbstractQoSDescription *aqd)

Private Attributes

- AudioReaderInterface * Source
- card8 * FrameBuffer
- cardinal FrameBufferPos
- cardinal FrameBufferSize
- card64 FramePosition
- card64 FrameMaxPosition
- AudioQuality FrameQualitySetting
- integer MediaInfoCounter
- card64 ByteRateLimit
- cardinal NetworkQualityDecrement
- cardinal SendError
- card8 ErrorCode

6.131.1 Detailed Description

Simple Audio Encoder.

This class is an simple audio encoder. It does no error correction or redundant transmission.

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

6.131.2 Constructor & Destructor Documentation

6.131.2.1 SimpleAudioEncoder::SimpleAudioEncoder (AudioReaderInterface * *audioReader*)

Constructor for the audio encoder.

Parameters

audioReader	AudioReaderInterface for the audio input.

6.131.2.2 SimpleAudioEncoder::~SimpleAudioEncoder ( )

Destructor.

6.131.3 Member Function Documentation

6.131.3.1 void SimpleAudioEncoder::activate ( ) [virtual]

activate() implementation of EncoderInterface.

See also

EncoderInterface::activate

Implements EncoderInterface.

6.131.3.2 bool SimpleAudioEncoder::checkInterval (card64 & *time,* bool & *newRUList*) [virtual]

checkInterval() implementation of EncoderInterface.

See also

EncoderInterface::checkInterval

Implements EncoderInterface.

6.131.3.3 void SimpleAudioEncoder::deactivate ( ) [virtual]

deactivate() implementation of EncoderInterface.

See also

EncoderInterface::deactivate

Implements EncoderInterface.

6.131.3.4 double SimpleAudioEncoder::getFrameRate ( ) const [virtual]

getFrameRate() implementation of EncoderInterface.

Returns

EncoderInterface::getFrameRate

Implements EncoderInterface.

6.131.3.5 cardinal SimpleAudioEncoder::getNextPacket (EncoderPacket * *encoderPacket*) [virtual]

getNextPacket() implementation of EncoderInterface.

See also

EncoderInterface::getNextPacket

Implements EncoderInterface.

6.131.3.6 AbstractQoSDescription * SimpleAudioEncoder::getQoSDescription (const cardinal *pktHeaderSize,* const cardinal *pktMaxSize,* const card64 *offset*) [virtual]

getQoSDescription() implementation of EncoderInterface.

See also

EncoderInterface::getQoSDescription

Implements EncoderInterface.

6.131.3.7 const card16 SimpleAudioEncoder::getTypeID ( ) const [virtual]

getTypeID() implementation of EncoderInterface.

See also

EncoderInterface::getTypeID

Implements EncoderInterface.

6.131.3.8 const char * SimpleAudioEncoder::getTypeName ( ) const [virtual]

getTypeName implementation of EncoderInterface.

See also

EncoderInterface::getTypeName

Implements EncoderInterface.

6.131.3.9 bool SimpleAudioEncoder::prepareNextFrame (const cardinal *headerSize,* const cardinal *maxPacketSize,* const cardinal *flags*) [virtual]

prepareNextFrame() implementation of EncoderInterface.

See also

EncoderInterface::prepareNextFrame

Implements EncoderInterface.

6.131.3.10 void SimpleAudioEncoder::reset ( ) [virtual]

reset() implementation of EncoderInterface.

See also

EncoderInterface::reset

Implements EncoderInterface.

6.131.3.11 void SimpleAudioEncoder::updateQuality (const AbstractQoSDescription * *aqd*) [virtual]

updateQuality() implementation of EncoderInterface.

See also

EncoderInterface::updateQuality

Implements EncoderInterface.

6.131.4 Member Data Documentation

6.131.4.1 card64 SimpleAudioEncoder::ByteRateLimit [private]

6.131.4.2 card8 SimpleAudioEncoder::ErrorCode [private]

6.131.4.3 card8* SimpleAudioEncoder::FrameBuffer [private]

6.131.4.4 cardinal SimpleAudioEncoder::FrameBufferPos [private]

6.131.4.5 cardinal SimpleAudioEncoder::FrameBufferSize [private]

6.131.4.6 card64 SimpleAudioEncoder::FrameMaxPosition [private]

6.131.4.7 card64 SimpleAudioEncoder::FramePosition [private]

6.131.4.8 AudioQuality SimpleAudioEncoder::FrameQualitySetting [private]

6.131.4.9 integer SimpleAudioEncoder::MediaInfoCounter [private]

6.131.4.10 cardinal SimpleAudioEncoder::NetworkQualityDecrement [private]

6.131.4.11 cardinal SimpleAudioEncoder::SendError [private]

6.131.4.12 AudioReaderInterface* SimpleAudioEncoder::Source [private]

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

- simpleaudioencoder.h
- simpleaudioencoder.cc

6.132 SimpleAudioPacket Struct Reference

Simple Audio Packet.

`#include <simpleaudiopacket.h>`

Public Types

- enum SimpleAudioFlags { SAF_Data = 0, SAF_MediaInfo = 1 }

Public Member Functions

- SimpleAudioPacket ()
- void translate ()
- void reset ()

Static Public Member Functions

- static AudioQuality calculateQualityForLimits (const AudioQualityInterface &userSetting, const AudioQualityInterface &inputQuality, const card64 totalByteRateLimit, const cardinal networkQualityDecrement, const cardinal headerSize, const cardinal maxPacketSize)
- static cardinal calculateFrameSize (const cardinal inputBytesPerSecond, const cardinal inputFrameSize)

Public Attributes

- card32 FormatID
- card16 SamplingRate
- card8 Channels
- card8 Bits
- card64 Position
- card64 MaxPosition
- card8 ErrorCode
- card8 Flags
- enum SimpleAudioPacket::SimpleAudioFlags __attribute__
- char Data []

Static Public Attributes

- static const card16 SimpleAudioTypeID = 0x2960
- static const char SimpleAudioTypeName [] = "Simple Audio Encoding"
- static const card32 SimpleAudioFormatID = 0x74660000 | SimpleAudioTypeID
- static const cardinal SimpleAudioMediaInfoPacketsPerSecond = 1
- static const cardinal SimpleAudioFramesPerSecond = 15
- static const cardinal SimpleAudioFrameSize = 2352 * 5
- static const cardinal SimpleAudioMaxTransferDelay = 1500 * 16
- static const cardinal SimpleAudioQualityLevels = AudioQuality::QualityLevels

6.132.1 Detailed Description

Simple Audio Packet.

This struct defines the packet format for the simple audio encoder.

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

See also

SimpleAudioEncoder
SimpleAudioDecoder

6.132.2 Member Enumeration Documentation

6.132.2.1 enum SimpleAudioPacket::SimpleAudioFlags

Emumeration of Flags.

Enumerator:

SAF_Data

SAF_MediaInfo

6.132.3 Constructor & Destructor Documentation

6.132.3.1 SimpleAudioPacket::SimpleAudioPacket ( )

Constructor.

6.132.4 Member Function Documentation

6.132.4.1 cardinal SimpleAudioPacket::calculateFrameSize (const cardinal *inputBytesPerSecond,* const cardinal *inputFrameSize*) [static]

Calculate output frame size from given input bytes per second and input frame size.

Parameters

inputBytesPerSecond	Input source's bytes per second.
inputFrameSize	Input source's frame size.

Returns

The calculated frame size.

6.132.4.2 AudioQuality SimpleAudioPacket::calculateQualityForLimits (const AudioQualityInterface & *userSetting,* const AudioQualityInterface & *inputQuality,* const card64 *totalByteRateLimit,* const cardinal *networkQualityDecrement,* const cardinal *headerSize,* const cardinal *maxPacketSize*) [static]

Quality calculation for given user quality limited by input quality, byte rate and network quality decrement with given header size (eg. IP + UDP + RTP) and maximum packet size.

Parameters

userSetting	User's quality setting.
inputQuality	Input source's quality.
byteRateLimit	Byte rate limit.
networkQualityDecrement	Number of steps for decrement of user's quality.
headerSize	Header size (eg. IP + UDP + RTP). SimpleAudioPacket size is added automatically.
maxPacketSize	Maximum packet size.

Returns

The calculated quality.

6.132.4.3 void SimpleAudioPacket::reset ( )

Reset report.

6.132.4.4 void SimpleAudioPacket::translate ( )

Translate byte order.

6.132.5 Member Data Documentation

6.132.5.1 enum SimpleAudioPacket::SimpleAudioFlags SimpleAudioPacket::__attribute__

6.132.5.2 card8 SimpleAudioPacket::Bits

Number of audio bits.

6.132.5.3 card8 SimpleAudioPacket::Channels

Number of audio channels.

6.132.5.4 char SimpleAudioPacket::Data[]

Packet data.

6.132.5.5 card8 SimpleAudioPacket::ErrorCode

Error code.

6.132.5.6 card8 SimpleAudioPacket::Flags

Flags.

6.132.5.7 card32 SimpleAudioPacket::FormatID

Packet format ID.

6.132.5.8 card64 SimpleAudioPacket::MaxPosition

Maximum position in nanoseconds.

6.132.5.9 card64 SimpleAudioPacket::Position

Current position in nanoseconds.

6.132.5.10 card16 SimpleAudioPacket::SamplingRate

Audio sampling rate.

6.132.5.11 const card32 SimpleAudioPacket::SimpleAudioFormatID = 0x74660000 | SimpleAudioTypeID [static]

Simple Audio Encoding package format ID.

6.132.5.12 const cardinal SimpleAudioPacket::SimpleAudioFrameSize = 2352 * 5 [static]

Simple Audio frame size.

6.132.5.13 const cardinal SimpleAudioPacket::SimpleAudioFramesPerSecond = 15 [static]

Simple Audio frames per second.

6.132.5.14 const cardinal SimpleAudioPacket::SimpleAudioMaxTransferDelay = 1500 * 16 [static]

Simple Audio maximum transfer delay.

6.132.5.15 const cardinal SimpleAudioPacket::SimpleAudioMediaInfoPacketsPerSecond = 1 [static]

Simple Audio MediaInfo packets per second.

6.132.5.16 const cardinal SimpleAudioPacket::SimpleAudioQualityLevels = AudioQuality::QualityLevels [static]

Simple Audio number of quality levels.

6.132.5.17 const card16 SimpleAudioPacket::SimpleAudioTypeID = 0x2960 [static]

Type ID for Simple Audio Encoding.

6.132.5.18 const char SimpleAudioPacket::SimpleAudioTypeName = "Simple Audio Encoding" [static]

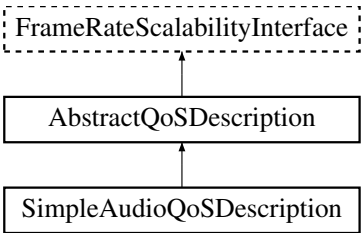
Name for Simple Audio Encoding.

The documentation for this struct was generated from the following files:

- simpleaudiopacket.h
- simpleaudiopacket.cc

6.133 SimpleAudioQoSDescription Class Reference

Inheritance diagram for SimpleAudioQoSDescription:



Public Member Functions

- SimpleAudioQoSDescription ()
- ~SimpleAudioQoSDescription ()
- void updateDescription (const cardinal pktHeaderSize, const cardinal pktMaxSize)
- cardinal getLayers () const
- AbstractLayerDescription * getLayer (const cardinal layer) const

6.133.1 Constructor & Destructor Documentation

6.133.1.1 SimpleAudioQoSDescription::SimpleAudioQoSDescription ( )

6.133.1.2 SimpleAudioQoSDescription::~SimpleAudioQoSDescription ( )

6.133.2 Member Function Documentation

6.133.2.1 AbstractLayerDescription * SimpleAudioQoSDescription::getLayer (const cardinal *layer*) const [virtual]

Get layer.

Parameters

layer	Layer number.

Returns

Layer.

Implements AbstractQoSDescription.

6.133.2.2 cardinal SimpleAudioQoSDescription::getLayers ( ) const [virtual]

Get number of layers.

Returns

Number of layers.

Implements AbstractQoSDescription.

6.133.2.3 void SimpleAudioQoSDescription::updateDescription (const cardinal *pktHeaderSize*, const cardinal *pktMaxSize*) [virtual]

Update description.

Parameters

pktHeader-Size	Packet header size.
pktMaxSize	Maximum packet size.

Implements AbstractQoSDescription.

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

- t1.cc

6.134 Socket Class Reference

Socket.

`#include <tdsocket.h>`

Public Types

- enum SocketFamily { UndefinedSocketFamily = -1, IP = 255, IPv4 = AF_INET, IPv6 = AF_INET6, Unix = AF_UNIX }

- enum SocketType { UndefinedSocketType = -1, UDP = SOCK_DGRAM, Datagram = SOCK_DGRAM, TCP = SOCK_STREAM, Stream = SOCK_STREAM, Raw = SOCK_RAW, RDM = SOCK_RDM, SeqPacket = SOCK_SEQPACKET }
- enum SocketProtocol { UndefinedSocketProtocol = -1, Default = 0, ICMPv4 = IPPROTO_ICMP, ICMPv6 = IPPROTO_ICMPV6, SCTP = IPPROTO_SCTP }
- enum GetSocketAddressFlags { GLAF_HideLoopback = (1 << 0), GLAF_HideLinkLocal = (1 << 1), GLAF_HideSiteLocal = (1 << 2), GLAF_HideLocal = GLAF_HideLoopback|GLAF_HideLinkLocal|GLAF_HideSiteLocal, GLAF_HideAnycast = (1 << 3), GLAF_HideMulticast = (1 << 4), GLAF_HideBroadcast = (1 << 5), GLAF_HideReserved = (1 << 6), GLAF_Default = GLAF_HideLoopback|GLAF_HideLinkLocal|Socket::GLAF_HideSiteLocal|GLAF_HideBroadcast|GLAF_HideMulticast|GLAF_HideAnycast }

Public Member Functions

- Socket ()
- Socket (const integer family, const integer socketType, const integer socketProtocol=Default)
- ~Socket ()
- bool create (const integer socketFamily=IP, const integer socketType=TCP, const integer socketProtocol=Default)
- void close ()
- void shutdown (const cardinal shutdownLevel)
- integer getFamily () const
- integer getType () const
- integer getProtocol () const
- bool ready () const
- bool bind (const SocketAddress &address=InternetAddress())
- bool bindx (const SocketAddress **addressArray=NULL, const cardinal addresses=0, const integer flags=0)
- bool listen (const cardinal backlog=5)
- Socket * accept (SocketAddress **address=NULL)
- bool connect (const SocketAddress &address, const card8 trafficClass=0)
- bool connectx (const SocketAddress **addressArray, const size_t addresses)
- integer getLastError ()
- integer getSocketOption (const cardinal level, const cardinal optionNumber, void *optionValue, socklen_t *optionLength)
- cardinal getSoLinger ()
- bool getSoReuseAddress ()
- bool getSoBroadcast ()
- bool getTCPNoDelay ()
- bool getBlockingMode ()
- integer setSocketOption (const cardinal level, const cardinal optionNumber, const void *optionValue, const socklen_t optionLength)
- bool setSoLinger (const bool on, const cardinal linger)
- bool setSoReuseAddress (const bool on)

- bool setSoBroadcast (const bool on)
- bool setTCPNoDelay (const bool on)
- bool setBlockingMode (const bool on)
- card32 getSendFlowLabel () const
- card8 getSendTrafficClass () const
- card32 getReceivedFlowLabel () const
- card8 getReceivedTrafficClass () const
- ssize_t send (const void *buffer, const size_t length, const integer flags=0, const card8 trafficClass=0x00)
- ssize_t sendTo (const void *buffer, const size_t length, const integer flags, const SocketAddress &receiver, const card8 trafficClass=0x00)
- ssize_t sendMsg (const struct msghdr *msg, const integer flags, const card8 trafficClass=0x00)
- ssize_t write (const void *buffer, const size_t length)
- ssize_t receive (void *buffer, const size_t length, integer &flags)
- ssize_t receiveFrom (void *buffer, const size_t length, SocketAddress &sender, integer &flags)
- ssize_t receiveMsg (struct msghdr *msg, const integer flags, const bool internalCall=false)
- ssize_t read (void *buffer, const size_t length)
- integer fcntl (const integer cmd, long arg)
- integer fcntl (const integer cmd, struct flock *lock)
- integer ioctl (const integer request, const void *argp)
- bool getSocketAddress (SocketAddress &address) const
- bool getPeerAddress (SocketAddress &address) const
- bool addMulticastMembership (const SocketAddress &address, const char *interface=NULL)
- bool dropMulticastMembership (const SocketAddress &address, const char *interface=NULL)
- bool getMulticastLoop ()
- bool setMulticastLoop (const bool on)
- card8 getMulticastTTL ()
- bool setMulticastTTL (const card8 ttl)
- InternetFlow allocFlow (const InternetAddress &address, const card32 flowLabel=0, const card8 shareLevel=2)
- void freeFlow (InternetFlow &flow)
- bool renewFlow (InternetFlow &flow, const cardinal expires, const cardinal linger=6)
- bool renewFlow (const cardinal expires, const cardinal linger=6)
- int getSystemSocketDescriptor () const

Static Public Member Functions

- static bool bindSocketPair (Socket &socket1, Socket &socket2, const SocketAddress &address=InternetAddress())

- static bool bindxSocketPair (Socket &socket1, Socket &socket2, const SocketAddress **addressArray=NULL, const cardinal addresses=0, const integer flags=0)
- static bool getLocalAddressList (SocketAddress **&addressList, cardinal &numberOfNets, const cardinal flags=GLAF_Default)

Static Public Attributes

- static const cardinal MinAutoSelectPort = 16384
- static const cardinal MaxAutoSelectPort = 61000

Private Member Functions

- void init ()
- bool setTypeOfService (const card8 trafficClass)
- ssize_t recvFrom (int fd, void *buf, const size_t len, integer &flags, struct sockaddr *addr, socklen_t *addrlen)
- bool multicastMembership (const SocketAddress &address, const char *interface, const bool add)
- void packSocketAddressArray (const sockaddr_storage *addrArray, const size_t addrs, sockaddr *packedArray)

Private Attributes

- int SocketDescriptor
- integer Family
- integer Type
- integer Protocol
- card32 SendFlow
- card32 ReceivedFlow
- integer LastError
- cardinal Backlog
- sockaddr * Destination

Friends

- class TrafficShaper

6.134.1 Detailed Description

Socket.

This class manages a socket. IPv6 support is automatically available, when supported by the system.

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

6.134.2 Member Enumeration Documentation

6.134.2.1 enum Socket::GetSocketAddressFlags

Enumerator:

GLAF_HideLoopback

GLAF_HideLinkLocal

GLAF_HideSiteLocal

GLAF_HideLocal

GLAF_HideAnycast

GLAF_HideMulticast

GLAF_HideBroadcast

GLAF_HideReserved

GLAF_Default

6.134.2.2 enum Socket::SocketFamily

Enumerator:

UndefinedSocketFamily

IP

IPv4

IPv6

Unix

6.134.2.3 enum Socket::SocketProtocol

Enumerator:

UndefinedSocketProtocol

Default

ICMPv4

ICMPv6

SCTP

6.134.2.4 enum Socket::SocketType

Enumerator:

UndefinedSocketType

UDP

Datagram

TCP

Stream

Raw

RDM

SeqPacket

6.134.3 Constructor & Destructor Documentation

6.134.3.1 Socket::Socket ( )

Constructor.

6.134.3.2 Socket::Socket (const integer *family,* const integer *socketType,* const integer *socketProtocol* = Default)

Constructor for a new socket. For automatic usage of IPv6 when available, set communication family to IP. Use IPv4/IPv6 only if a special protocol version is necessary! The creation success can be checked using ready() method.

Parameters

family	Communication family (e.g. IP).
socketType	Socket type (e.g. TCP, UDP).
socketProtocol	Socket protocol (e.g. Default).

See also

ready

6.134.3.3 Socket::~Socket ( )

Destructor.

6.134.4 Member Function Documentation

6.134.4.1 Socket * Socket::accept (SocketAddress ** *address* = NULL)

Accept a connection.

Parameters

address	Reference to store SocketAddress object to with peer's address to (NULL to skip).

Returns

New socket.

6.134.4.2 bool Socket::addMulticastMembership (const SocketAddress & *address*, const char * *interface* = NULL) [inline]

Add multicast membership.

Parameters

address	Multicast address.
interface	Interface name.

Returns

true for success; false otherwise.

6.134.4.3 InternetFlow Socket::allocFlow (const InternetAddress & *address*, const card32 *flowLabel* = 0, const card8 *shareLevel* = 2)

Allocate a new flow to a given destination. A InternetFlow object is returned, the value flow.getFlowLabel() will not be 0, if the allocFlow() call was successful.

Parameters

address	Address of the destination.
flowLabel	Flowlabel; 0 for random value.
shareLevel	Share level for flow label.

Returns

InternetFlow.

6.134.4.4 bool Socket::bind (const SocketAddress & *address* = InternetAddress ())

Bind socket to given address. If address is null address, then INADDR_ANY and an automatically selected port will be used.

Parameters

address	Socket address.

Returns

true on success; false otherwise.

6.134.4.5 bool Socket::bindSocketPair (Socket & *socket1*, Socket & *socket2*, const SocketAddress & *address* = InternetAddress ()) [static]

Bind a pair of sockets to a given address and port number x and x + 1. x will be a random number, if given port number is 0.

Parameters

socket1	First socket.
socket2	Second socket.
address	Address (e.g ipv6-localhost:0) or NULL for Any address.

See also

bind

6.134.4.6 bool Socket::bindx (const SocketAddress ** *addressArray* = NULL, const cardinal *addresses* = 0, const integer *flags* = 0)

O * Bind socket to one or more given addresses. If no addresses are given, INADDR_ANY and an automatically selected port will be used.

Parameters

addressArray	Array of socket addresses.
addresses	Number of addresses.
flags	Flags.

Returns

true on success; false otherwise.

6.134.4.7 bool Socket::bindxSocketPair (Socket & *socket1*, Socket & *socket2*, const SocketAddress ** *addressArray* = NULL, const cardinal *addresses* = 0, const integer *flags* = 0) [static]

Bind a pair of sockets to a given address and port number x and x + 1. x will be a random number, if given port number is 0. If no addresses are given, INADDR_ANY and an automatically selected port will be used.

Parameters

socket1	First socket.
socket2	Second socket.
addressArray	Array of socket addresses.
addresses	Number of addresses.
flags	bindx() flags.

See also

bindx

6.134.4.8 void Socket::close ( )

Close socket.

6.134.4.9 bool Socket::connect (const SocketAddress & *address*, const card8 *trafficClass* = 0)

Connect socket to given address. A value for traffic class is supported if the connection is an IPv6 connection; otherwise it is ignored.

Parameters

address	Address.
trafficClass	Traffic class of the connection (IPv6 only!)

Returns

true on success; false otherwise.

6.134.4.10 bool Socket::connectx (const SocketAddress ** *addressArray*, const size_t *addresses*)

Connect socket to destination given by list of addresses. A value for traffic class is supported if the connection is an IPv6 connection; otherwise it is ignored.

Parameters

address	Address.
trafficClass	Traffic class of the connection (IPv6 only!)

Returns

true on success; false otherwise.

6.134.4.11 bool Socket::create (const integer *socketFamily* = IP, const integer *socketType* = TCP, const integer *socketProtocol* = Default)

Close existing socket and create new socket. For automatic usage of IPv6 when available, set communication family to IP. Use IPv4/IPv6 only if a special protocol version is necessary!

Parameters

socketFamily	Communication family (e.g. IP).
socketType	Socket type (e.g. TCP, UDP).
socket-Protocol	Socket protocol (e.g. Default).

Returns

true, if creation was sucessful; false otherwise.

6.134.4.12 bool Socket::dropMulticastMembership (const SocketAddress & *address*, const char * *interface* = NULL) [inline]

Drop multicast membership.

Parameters

address	Multicast address.
interface	Interface name.

Returns

true for success; false otherwise.

6.134.4.13 integer Socket::fcntl (const integer *cmd*, long *arg*) [inline]

Wrapper for fcntl().

Parameters

cmd	Command.
arg	Argument.

Returns

Result of fcntl() call.

6.134.4.14 integer Socket::fcntl (const integer *cmd*, struct flock * *lock*) [inline]

Wrapper for fcntl().

Parameters

cmd	Command.
lock	Lock.

Returns

Result of fcntl() call.

6.134.4.15 void Socket::freeFlow (InternetFlow & *flow*)

Free a flow.

Parameters

flow	Flow to be freed.

6.134.4.16 bool Socket::getBlockingMode ( )

Check, if blocking mode is on.

Returns

true, if blocking mode is on; false otherwise.

6.134.4.17 integer Socket::getFamily ( ) const [inline]

Get socket's family.

Returns

Socket family.

6.134.4.18 integer Socket::getLastError ( ) [inline]

Get last error code. It will be reset to 0 after copying.

Returns

Last error code.

6.134.4.19 bool Socket::getLocalAddressList (SocketAddress **& *addressList*, cardinal & *numberOfNets*, const cardinal *flags* = GLAF_Default) [static]

Get list of all local addresses (IPv4 and IPv6 are currently supported). The resulting list has to be deallocated using SocketAddress::deleteAddressList().

Parameters

addressList	Reference to store address list to.
numberOfNets	Reference to store number of addresses to.
flags	Flags.

Returns

true for success; false otherwise.

See also

SocketAddress::deleteAddressList

6.134.4.20 bool Socket::getMulticastLoop ( )

Get multicast loop mode.

Returns

true if multicast loop is enabled, false otherwise.

6.134.4.21 card8 Socket::getMulticastTTL ( )

Get multicast TTL.

Returns

Multicast TTL.

6.134.4.22 bool Socket::getPeerAddress (SocketAddress & *address*) const

Get the peer's address. Note: A socket has to be connected to a peer first to get a peer address!

Parameters

address	Reference to SocketAddress to write address to.

Returns

true, if call was successful; false otherwise.

See also

bind
connect
getSocketAddress

6.134.4.23 integer Socket::getProtocol ( ) const [inline]

Get socket's protocol.

Returns

Socket protocol.

6.134.4.24 card32 Socket::getReceivedFlowLabel ( ) const [inline]

Get last received flow label.

Returns

Last received flow label or 0, if there is no flow label.

6.134.4.25 card8 Socket::getReceivedTrafficClass ( ) const [inline]

Get last received traffic class.

Returns

Last received traffic class or 0, if there is no traffic class.

6.134.4.26 card32 Socket::getSendFlowLabel ( ) const [inline]

Get flow label of the connection.

Returns

Flow label of the connection or 0, if there is no flow label.

See also

connect

6.134.4.27 card8 Socket::getSendTrafficClass ( ) const [inline]

Get traffic class of the connection.

Returns

Traffic class of the connection or 0, if there is no traffic class.

See also

connect

6.134.4.28 bool Socket::getSoBroadcast ( )

Get SO_BROADCAST option of socket.

Returns

SO_BROADCAST value.

6.134.4.29 bool Socket::getSocketAddress (SocketAddress & *address*) const

Get the socket's address. Note: A socket has to be bound to an address and port or connected to a peer first to let the socket have an address!

Parameters

address	Reference to SocketAddress to write address to.

Returns

true, if call was successful; false otherwise.

See also

bind
connect
getPeerAddress

6.134.4.30 integer Socket::getSocketOption (const cardinal *level*, const cardinal *optionNumber*, void * *optionValue*, socklen_t * *optionLength*) [inline]

Get socket option (wrapper for getsockopt());

Parameters

level	Level (e.g. SOL_SOCKET).
optionNumber	Option (e.g. SO_REUSEADDR).
optionValue	Memory to store option got from getsockopt().
optionLength	Memory with size of option memory.

Returns

Result from getsockopt().

6.134.4.31 cardinal Socket::getSoLinger ( )

Get SO_LINGER option of socket.

Returns

SO_LINGER value.

6.134.4.32 bool Socket::getSoReuseAddress ( )

Get SO_REUSEADDR option of socket.

Returns

SO_REUSEADDR value.

6.134.4.33 int Socket::getSystemSocketDescriptor ( ) const [inline]

Get system's socket descriptor. Warning: It is not recommended to manipulate the socket directly. Use Socket's methods instead.

Returns

Socket descriptor.

6.134.4.34 bool Socket::getTCPNoDelay ( )

Get TCP_NODELAY option of socket.

Returns

TCP_NODELAY value.

6.134.4.35 integer Socket::getType ( ) const [inline]

Get socket's type.

Returns

Socket type.

6.134.4.36 void Socket::init ( ) [private]

6.134.4.37 integer Socket::ioctl (const integer *request*, const void * *argp*) [inline]

Wrapper for ioctl().

Parameters

request	Request.
argp	Argument.

Returns

Result of ioctl() call.

6.134.4.38 bool Socket::listen (const cardinal *backlog* = 5)

Set socket to listen mode with given backlog (queue length for sockets waiting for acception).

Parameters

backlog	Backlog.

Returns

true on success; false otherwise.

6.134.4.39 bool Socket::multicastMembership (const SocketAddress & *address,* const char * *interface,* const bool *add*) [private]

6.134.4.40 void Socket::packSocketAddressArray (const sockaddr_storage * *addrArray,* const size_t *addrs,* sockaddr * *packedArray*) [private]

6.134.4.41 ssize_t Socket::read (void * *buffer,* const size_t *length*) [inline]

Wrapper for read().

Parameters

buffer	Buffer to read data to.
length	Maximum length of data to be received.

Returns

Bytes read or error code < 0.

6.134.4.42 bool Socket::ready ( ) const [inline]

Check, if socket is ready.

Returns

true, if socket is ready; false otherwise.

6.134.4.43 ssize_t Socket::receive (void * *buffer,* const size_t *length,* integer & *flags*) [inline]

Wrapper for recv().

Parameters

buffer	Buffer to read data to.
length	Maximum length of data to be received.
flags	Flags for recvmsg().

Returns

Bytes read or error code < 0.

6.134.4.44 ssize_t Socket::receiveFrom (void * *buffer,* const size_t *length,* SocketAddress & *sender,* integer & *flags*)

Wrapper for recvfrom().

Parameters

buffer	Buffer to receive data to.
length	Maximum length of data to be received.
sender	Address to store sender's address.
flags	Flags for recvmsg().

Returns

Bytes received or error code < 0.

6.134.4.45 ssize_t Socket::receiveMsg (struct msghdr * *msg,* const integer *flags,* const bool *internalCall* = `false`)

Wrapper for recvmsg().

Parameters

msg	Message.
flags	Flags for recvmsg().
internalCall	Internal usage only; set to false.

Returns

Result of recvmsg() call.

6.134.4.46 ssize_t Socket::recvFrom (int *fd,* void * *buf,* const size_t *len,* integer & *flags,* struct sockaddr * *addr,* socklen_t * *addrlen*) `[private]`

6.134.4.47 bool Socket::renewFlow (InternetFlow & *flow,* const cardinal *expires,* const cardinal *linger* = `6`)

Renew a flow label allocation with given expires and linger (default 6) values. The expires value gives the seconds to go until the flow label expires, the linger value gives the timeout in seconds the freed flow label cannot be allocated again.

Parameters

flow	Flow to be renewed.
expires	Seconds until the flow label expires.
linger	Linger (default 6).

Returns

true on success; false otherwise.

6.134.4.48 bool Socket::renewFlow (const cardinal *expires*, const cardinal *linger* = 6)

Renew current flow's flow label allocation with given expires and linger (default 6) values.

Parameters

expires	Seconds until the flow label expires.
linger	Linger (default 6).

Returns

true on success; false otherwise.

6.134.4.49 ssize_t Socket::send (const void * *buffer*, const size_t *length*, const integer *flags* = 0, const card8 *trafficClass* = 0x00)

Wrapper for send(). send() will set the packet's traffic class, if trafficClass is not 0. In this case, the packet will be sent by sendto() to the destination address, the socket is connected to!

Parameters

buffer	Buffer with data to send.
length	Length of data to send.
flags	Flags for sendto().
trafficClass	Traffic class for packet.

Returns

Bytes sent or error code < 0.

6.134.4.50 ssize_t Socket::sendMsg (const struct msghdr * *msg*, const integer *flags*, const card8 *trafficClass* = 0x00)

Wrapper for sendmsg().

Parameters

msg	Message.
flags	Flags.
trafficClass	Traffic class for packet.

Returns

Result of sendmsg() call.

6.134.4.51 ssize_t Socket::sendTo (const void * *buffer,* const size_t *length,* const integer *flags,* const SocketAddress & *receiver,* const card8 *trafficClass* = 0x00)

Wrapper for sendto(). sendto() will set the packet's traffic class, if trafficClass is not 0.

Parameters

buffer	Buffer with data to send.
length	Length of data to send.
flags	Flags for sendto().
receiver	Address of receiver.
trafficClass	Traffic class for packet.

Returns

Bytes sent or error code < 0.

6.134.4.52 bool Socket::setBlockingMode (const bool *on*)

Set blocking mode.

Parameters

on	True to set blocking mode, false to unset.
true	for success; false otherwise.

Returns

true for success; false otherwise.

6.134.4.53 bool Socket::setMulticastLoop (const bool *on*)

Set multicast loop mode.

Parameters

on	true to enable, false to disable.

Returns

true for success; false otherwise.

6.134.4.54 bool Socket::setMulticastTTL (const card8 *ttl*)

Set multicast TTL.

Parameters

ttl	TTL.

Returns

true for success; false otherwise.

6.134.4.55 bool Socket::setSoBroadcast (const bool *on*)

Set SO_BROADCAST option of socket.

Parameters

on	true to set SO_BROADCAST on; false otherwise.

Returns

true for success; false otherwise.

6.134.4.56 integer Socket::setSocketOption (const cardinal *level,* const cardinal *optionNumber,* const void * *optionValue,* const socklen_t *optionLength*) [inline]

Get socket option (wrapper for getsockopt());

Parameters

level	Level (e.g. SOL_SOCKET).
option-Number	Option (e.g. SO_REUSEADDR).
optionValue	Memory with option.
option-Length	Length of option memory.

Returns

Result from setsockopt().

6.134.4.57 bool Socket::setSoLinger (const bool *on,* const cardinal *linger*)

Set SO_LINGER option of socket.

Parameters

on	true to set linger on; false otherwise.
linger	SO_LINGER in seconds.

Returns

true for success; false otherwise.

6.134.4.58 bool Socket::setSoReuseAddress (const bool *on*)

Set SO_REUSEADDR option of socket.

Parameters

on	true to set SO_REUSEADDR on; false otherwise.

Returns

true for success; false otherwise.

6.134.4.59 bool Socket::setTCPNoDelay (const bool *on*)

Set TCP_NODELAY option of socket.

Parameters

on	true to set TCP_NODELAY on; false otherwise.

Returns

true for success; false otherwise.

6.134.4.60 bool Socket::setTypeOfService (const card8 *trafficClass*) [private]

6.134.4.61 void Socket::shutdown (const cardinal *shutdownLevel*)

Shutdown full-duplex connection partial or completely. SHUT_RD - further receives will be disallowed. SHUT_WR - further sends will be disallowed. SHUT_RDWR - further sends and receives will be disallowed.

Parameters

shutdown-Level	SHUT_RD, SHUT_WR, SHUT_RDWR.

6.134.4.62 ssize_t Socket::write (const void * *buffer,* const size_t *length*) [inline]

Wrapper for write().

Parameters

buffer	Buffer with data to write
length	Length of data to write

Returns

Bytes sent or error code < 0.

6.134.5 Friends And Related Function Documentation

6.134.5.1 friend class TrafficShaper [friend]

6.134.6 Member Data Documentation

6.134.6.1 cardinal Socket::Backlog [private]

6.134.6.2 sockaddr* Socket::Destination [private]

6.134.6.3 integer Socket::Family [private]

6.134.6.4 integer Socket::LastError [private]

6.134.6.5 const cardinal Socket::MaxAutoSelectPort = 61000 [static]

Maximum port number for bind()'s automatic port selection.

See also

bind

6.134.6.6 const cardinal Socket::MinAutoSelectPort = 16384 [static]

Minimum port number for bind()'s automatic port selection.

See also

bind

6.134.6.7 integer Socket::Protocol [private]

6.134.6.8 card32 Socket::ReceivedFlow [private]

6.134.6.9 card32 Socket::SendFlow [private]

6.134.6.10 int Socket::SocketDescriptor [private]

6.134.6.11 integer Socket::Type [private]

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

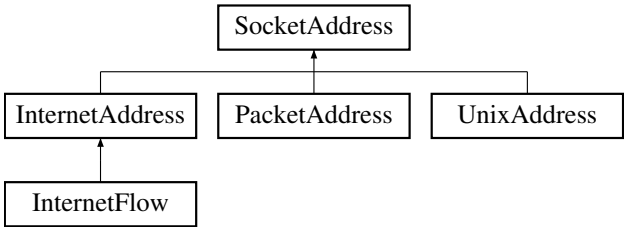
- tdsocket.h
- tdsocket.cc

6.135 SocketAddress Class Reference

Socket Address.

`#include <socketaddress.h>`

Inheritance diagram for SocketAddress:



Public Types

- enum PrintFormat { PF_Address = (1 << 0), PF_Hostname = (1 << 1), PF_Full = (PF_Address | PF_Hostname), PF_HidePort = (1 << 15), PF_Legacy = (1 << 16), PF_Default = (PF_Address | PF_Legacy) }

Public Member Functions

- virtual ~SocketAddress ()
- virtual SocketAddress * duplicate () const =0
- virtual void reset ()=0
- virtual bool isValid () const =0
- cardinal getPrintFormat () const
- void setPrintFormat (const cardinal format)
- virtual card16 getPort () const =0
- virtual void setPort (const card16 port)=0
- virtual integer getFamily () const =0
- virtual String getAddressString (const cardinal format=PF_Default) const =0
- virtual cardinal getSystemAddress (sockaddr *buffer, const socklen_t length, const cardinal type=AF_UNSPEC) const =0
- virtual bool setSystemAddress (const sockaddr *address, const socklen_t length)=0

Static Public Member Functions

- static SocketAddress * getLocalAddress (const SocketAddress &peer)
- static SocketAddress * createSocketAddress (const integer family)
- static SocketAddress * createSocketAddress (const cardinal flags, const String &name)
- static SocketAddress * createSocketAddress (const cardinal flags, const String &name, const card16 port)
- static SocketAddress * createSocketAddress (const cardinal flags, sockaddr *address, const socklen_t length)
- static SocketAddress ** newAddressList (const cardinal entries)
- static void deleteAddressList (SocketAddress **&addressArray)

Static Public Attributes

- static const cardinal MaxSockLen

Protected Attributes

- cardinal Format

Friends

- std::ostream & operator<< (std::ostream &os, const SocketAddress &sa)

6.135.1 Detailed Description

Socket Address.

This class is an interface for a socket address.

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

6.135.2 Member Enumeration Documentation

6.135.2.1 enum SocketAddress::PrintFormat

setPrintFormat() printing formats.

Enumerator:

PF_Address Print address.

PF_Hostname Print hostname, if possible. Otherwise, print address.

PF_Full Print both, address and hostname (if resolvable).

PF_HidePort Hide port number.

PF_Legacy Legacy mode: Do ∗not∗ print IPv4 addresses on IPv6-capable hosts as IPv4-mapped address. That is, 1.2.3.4 instead of ::ffff:1.2.3.4.

PF_Default Default print format.

6.135.3 Constructor & Destructor Documentation

6.135.3.1 SocketAddress::∼SocketAddress ( ) [virtual]

Destructor.

6.135.4 Member Function Documentation

6.135.4.1 SocketAddress ∗ SocketAddress::createSocketAddress (const integer *family*) [static]

Create SocketAddress object for given address family.

Parameters

family	Address family (e.g. AF_INET or AF_UNIX).

Returns

SocketAddress object or NULL in case of failure.

6.135.4.2 SocketAddress ∗ SocketAddress::createSocketAddress (const cardinal *flags*, const String & *name*) [static]

Create SocketAddress object from address string.

Parameters

flags	flags
string	Address string.

Returns

SocketAddress object or NULL in case of failure.

6.135.4.3 SocketAddress * SocketAddress::createSocketAddress (const cardinal *flags*, const String & *name*, const card16 *port*) [static]

Create SocketAddress object from address string and port number.

Parameters

flags	flags
string	Address string.
port	Port number.

Returns

SocketAddress object or NULL in case of failure.

6.135.4.4 SocketAddress * SocketAddress::createSocketAddress (const cardinal *flags*, sockaddr * *address*, const socklen_t *length*) [static]

Create SocketAddress object from system's sockaddr structure.

Parameters

flags	flags.
address	sockaddr.
length	Length of sockaddr.

Returns

SocketAddress object or NULL in case of failure.

6.135.4.5 void SocketAddress::deleteAddressList (SocketAddress **& *addressArray*) [static]

Deallocate NULL-terminated list of SocketAddress objects.

Parameters

address-Array	Address list.

See also

getLocalAddresses
accept

6.135.4.6 virtual SocketAddress* SocketAddress::duplicate ( ) const [pure virtual]

Create a duplicate of the SocketAddress object.

Returns

Copy of SocketAddress object.

Implemented in InternetAddress, PacketAddress, UnixAddress, and InternetFlow.

6.135.4.7 virtual String SocketAddress::getAddressString (const cardinal *format* = PF_Default) const [pure virtual]

Get address string.

Returns

Address string.

Implemented in InternetAddress, PacketAddress, UnixAddress, and InternetFlow.

6.135.4.8 virtual integer SocketAddress::getFamily ( ) const [pure virtual]

Get family of address.

Returns

Address family (e.g. AF_INET or AF_UNIX).

Implemented in InternetAddress, PacketAddress, and UnixAddress.

6.135.4.9 SocketAddress * SocketAddress::getLocalAddress (const SocketAddress & *peer*) [static]

Get the local host address. The parameter peer gives the address of the other host.

Parameters

peer	Address of peer.

Returns

Local SocketAddress or NULL in case of an error.

Examples: localhost => localhost address (127.0.0.1 or ::1). ethernet-host => ethernet interface address. internet-address => dynamic-ip address set by pppd.

6.135.4.10 virtual card16 SocketAddress::getPort ( ) const [pure virtual]

Get port of address.

Returns

Port.

Implemented in InternetAddress, PacketAddress, and UnixAddress.

6.135.4.11 cardinal SocketAddress::getPrintFormat ( ) const [inline]

Get printing format.

Returns

Print format.

6.135.4.12 virtual cardinal SocketAddress::getSystemAddress (sockaddr * *buffer*, const socklen_t *length*, const cardinal *type* = AF_UNSPEC) const [pure virtual]

Get system's sockaddr structure for the address.

Parameters

buffer	Buffer to write sockaddr to.
length	Length of buffer.
type	Socket address type, e.g. AF_INET or AF_INET6.

Returns

Length of written sockaddr structure.

Implemented in InternetAddress, PacketAddress, UnixAddress, and InternetFlow.

6.135.4.13 virtual bool SocketAddress::isValid ( ) const [pure virtual]

Check, if address is valid.

Returns

true, if address is valid; false otherwise.

Implemented in InternetAddress, PacketAddress, and UnixAddress.

6.135.4.14 SocketAddress ** SocketAddress::newAddressList (const cardinal *entries*) [static]

Allocate memory for NULL-terminated SocketAddress list with given number of entries.

Parameters

entries	Number of entries.

Returns

Address list or NULL, if out of memory.

6.135.4.15 virtual void SocketAddress::reset ( ) [pure virtual]

Reset address.

Implemented in InternetAddress, PacketAddress, UnixAddress, and InternetFlow.

6.135.4.16 virtual void SocketAddress::setPort (const card16 *port*) [pure virtual]

Set port of address.

Implemented in InternetAddress, PacketAddress, and UnixAddress.

6.135.4.17 void SocketAddress::setPrintFormat (const cardinal *format*) [inline]

Set printing format.

Parameters

format	Print format.

6.135.4.18 virtual bool SocketAddress::setSystemAddress (const sockaddr * *address*, const socklen_t *length*) [pure virtual]

Initialize the socket address from the system's sockaddr structure.

Parameters

address	sockaddr.
length	Length of sockaddr.

Implemented in InternetAddress, PacketAddress, and UnixAddress.

6.135.5 Friends And Related Function Documentation

6.135.5.1 std::ostream& operator<< (std::ostream & *os,* const SocketAddress & *sa*) [friend]

Output operator.

6.135.6 Member Data Documentation

6.135.6.1 cardinal SocketAddress::Format [protected]

Print format.

6.135.6.2 const cardinal SocketAddress::MaxSockLen [static]

Initial value:

```
(sizeof(sockaddr_un) > sizeof(sockaddr_storage)) ? sizeof(sockaddr_un) :
sizeof(sockaddr_storage)
```

Maximum sockaddr length in bytes.

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

- socketaddress.h
- socketaddress.cc

6.136 SocketMessage< size > Class Template Reference

Socket Message.

`#include <tdmessage.h>`

Public Member Functions

- SocketMessage ()
- void clear ()
- SocketAddress * getAddress () const
- void setAddress (const SocketAddress &address, const integer family=AF_UNSPEC)
- void setBuffer (void *buffer, const size_t buffersize)
- void * addHeader (const size_t payloadLength, const int level, const int type)
- cmsghdr * getFirstHeader ()
- cmsghdr * getNextHeader (cmsghdr *prev)
- int getFlags () const
- void setFlags (const int flags)

Public Attributes

- msghdr Header
- sockaddr_storage Address
- struct iovec IOVector
- char Control [CMSG_SPACE(size)]

Private Attributes

- cmsghdr * NextMsg

6.136.1 Detailed Description

template<const size_t size>class SocketMessage< size >

Socket Message.

This template class manages manages message structures used by sendmsg() and recvmsg(). The template parameter gives the size of the control data block.

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

6.136.2 Constructor & Destructor Documentation

6.136.2.1 template<const size_t size> SocketMessage< size >::SocketMessage ( ) [inline]

Constructor.

6.136.3 Member Function Documentation

6.136.3.1 template<const size_t size> void* SocketMessage< size >::addHeader (const size_t *payloadLength*, const int *level*, const int *type*) [inline]

Add control header of given cmsg level and type. Returns NULL, if there is not enough free space in the control data block. The new control header is cleared (i.e. all bytes set to 0).

Parameters

payload	Size of payload.
level	Level (e.g. IPPROTO_SCTP).
type	Type (e.g. SCTP_INIT).

Returns

Pointer to begin of ∗payload∗ area.

6.136.3.2 template<const size_t size> void SocketMessage< size >::clear ( ) [inline]

Clear structure.

6.136.3.3 template<const size_t size> SocketAddress∗ SocketMessage< size >::getAddress ( ) const [inline]

Get address as SocketAddress object. Note: This address has to be freed using delete operator!

Returns

SocketAddress object.

6.136.3.4 template<const size_t size> cmsghdr∗ SocketMessage< size >::getFirstHeader ( ) [inline]

Get first cmsg header in control block.

Returns

First cmsg header or NULL, if there are none.

6.136.3.5 template<const size_t size> int SocketMessage< size >::getFlags ( ) const [inline]

Get flags.

Returns

Flags.

6.136.3.6 template<const size_t size> cmsghdr∗ SocketMessage< size >::getNextHeader (cmsghdr ∗ *prev*) [inline]

Get next cmsg header in control block.

Parameters

prev	Previous cmsg header.

Returns

Next cmsg header or NULL, if there are no more.

6.136.3.7 template<const size_t size> void SocketMessage< size >::setAddress (const SocketAddress & *address,* const integer *family* = AF_UNSPEC) [inline]

Set address.

Parameters

address	SocketAddress object.
type	Address type (AF_UNSPEC to take default from address).

6.136.3.8 template<const size_t size> void SocketMessage< size >::setBuffer (void * *buffer,* const size_t *buffersize*) [inline]

Set buffer.

Parameters

buffer	Buffer.
bufferSize	Size of buffer.

6.136.3.9 template<const size_t size> void SocketMessage< size >::setFlags (const int *flags*) [inline]

Set flags.

Parameters

flags	Flags.

6.136.4 Member Data Documentation

6.136.4.1 template<const size_t size> sockaddr_storage SocketMessage< size >::Address

Storage for address.

6.136.4.2 template<const size_t size> char SocketMessage< size >::Control[CMSG_SPACE(size)]

Control data block, its size is given by the template parameter.

6.136.4.3 template<const size_t size> msghdr SocketMessage< size >::Header

msghdr structure.

6.136.4.4 template<const size_t size> struct iovec SocketMessage< size >::IOVector

iovec structure.

6.136.4.5 template<const size_t size> cmsghdr* SocketMessage< size >::NextMsg [private]

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

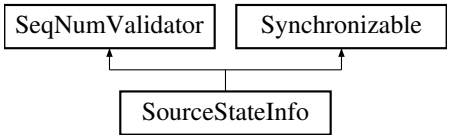
- tdmessage.h

6.137 SourceStateInfo Class Reference

Source State Info.

`#include <sourcestateinfo.h>`

Inheritance diagram for SourceStateInfo:



Public Member Functions

- SourceStateInfo ()
- SourceStateInfo & operator= (const SourceStateInfo &original)
- void reset ()
- card32 getSSRC () const
- card32 getLSR () const
- card32 calculateDLSR () const
- void setLSR (const card32 lsr)
- void setSSRC (card32 ssrc)

Private Attributes

- card64 LSRUpdateTimeStamp
- card32 ReceivedPrior

- card32 ExpectedPrior
- card32 FractionLost
- card32 SSRC
- card32 LSR

6.137.1 Detailed Description

Source State Info.

This class manages the source state information of an RTP receiver to be transmitted by a RTCPSender. See also RFC 1889 for more information on RTP.

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

See also

RTPReceiver
RTCPSender

6.137.2 Constructor & Destructor Documentation

6.137.2.1 SourceStateInfo::SourceStateInfo ( )

Constructor.

6.137.3 Member Function Documentation

6.137.3.1 card32 SourceStateInfo::calculateDLSR ( ) const

Calculate delay since last sender report time stamp using current time.

Returns

DLSR.

6.137.3.2 card32 SourceStateInfo::getLSR ( ) const [inline]

Get last sender report time stamp.

Returns

LSR.

6.137.3.3 card32 SourceStateInfo::getSSRC ( ) const [inline]

Get SSRC.

Returns

SSRC.

6.137.3.4 SourceStateInfo & SourceStateInfo::operator= (const SourceStateInfo & *original*)

Copy operation.

6.137.3.5 void SourceStateInfo::reset ( )

Reset.

Reimplemented from SeqNumValidator.

6.137.3.6 void SourceStateInfo::setLSR (const card32 *lsr*) [inline]

Set last sender report time stamp.

Parameters

lsr	LSR.

6.137.3.7 void SourceStateInfo::setSSRC (card32 *ssrc*) [inline]

Set SSRC.

Returns

SSRC.

6.137.4 Member Data Documentation

6.137.4.1 card32 SourceStateInfo::ExpectedPrior [private]

Reimplemented from SeqNumValidator.

6.137.4.2 card32 SourceStateInfo::FractionLost [private]

Reimplemented from SeqNumValidator.

6.137.4.3 card32 SourceStateInfo::LSR [private]

6.137.4.4 card64 SourceStateInfo::LSRUpdateTimeStamp [private]

6.137.4.5 card32 SourceStateInfo::ReceivedPrior [private]

Reimplemented from SeqNumValidator.

6.137.4.6 card32 SourceStateInfo::SSRC [private]

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

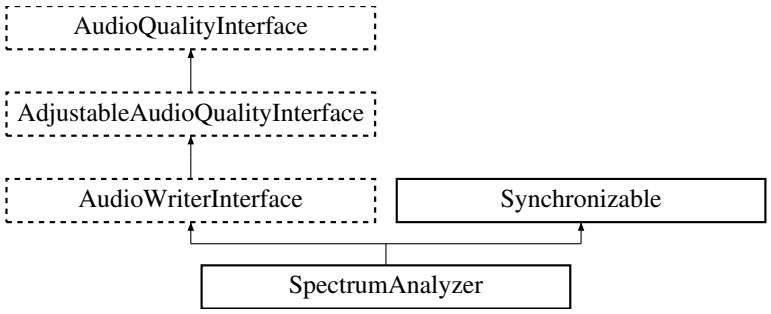
- sourcestateinfo.h
- sourcestateinfo.cc

6.138 SpectrumAnalyzer Class Reference

Spectrum Analyzer.

`#include <spectrumanalyzer.h>`

Inheritance diagram for SpectrumAnalyzer:



Public Member Functions

- SpectrumAnalyzer ()
- ~SpectrumAnalyzer ()
- card16 getSamplingRate () const
- card8 getBits () const
- card8 getChannels () const
- card16 getByteOrder () const
- cardinal getBytesPerSecond () const
- cardinal getBitsPerSample () const
- card16 setSamplingRate (const card16 samplingRate)
- card8 setBits (const card8 bits)

- card8 setChannels (const card8 channels)
- card16 setByteOrder (const card16 byteOrder)
- bool ready () const
- void sync ()
- bool write (const void *data, const size_t length)
- bool getSpectrum (cardinal *left, cardinal *right, const cardinal bars)

Private Member Functions

- void doFourierTransformation (card16 *data, cardinal *output, cardinal bars)

Private Attributes

- FastFourierTransformation * FFT
- cardinal InputBufferPos
- char InputBuffer [4 *FFTPoints]
- card16 AudioSamplingRate
- card8 AudioBits
- card8 AudioChannels
- card16 AudioByteOrder

Static Private Attributes

- static const cardinal FFTPoints = 256

6.138.1 Detailed Description

Spectrum Analyzer.

This class implements a spectrum analyzer device implementing AudioWriterInterface.

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

6.138.2 Constructor & Destructor Documentation

6.138.2.1 SpectrumAnalyzer::SpectrumAnalyzer ( )

Constructor.

6.138.2.2 SpectrumAnalyzer::~SpectrumAnalyzer ( )

Destructor.

6.138.3 Member Function Documentation

6.138.3.1 void SpectrumAnalyzer::doFourierTransformation (card16 * *data*, cardinal * *output*, cardinal *bars*) [private]

6.138.3.2 card8 SpectrumAnalyzer::getBits ( ) const [virtual]

getBits() Implementation of AudioQualityInterface.

See also

AudioQualityInterface::getBits

Implements AudioQualityInterface.

6.138.3.3 cardinal SpectrumAnalyzer::getBitsPerSample ( ) const [virtual]

getBitsPerSample() Implementation of AudioQualityInterface.

See also

AudioQualityInterface::getBitsPerSample

Implements AudioQualityInterface.

6.138.3.4 card16 SpectrumAnalyzer::getByteOrder ( ) const [virtual]

getByteOrder() Implementation of AudioQualityInterface.

See also

AudioQualityInterface::getByteOrder

Implements AudioQualityInterface.

6.138.3.5 cardinal SpectrumAnalyzer::getBytesPerSecond ( ) const [virtual]

getBytesPerSecond() Implementation of AudioQualityInterface.

See also

AudioQualityInterface::getBytesPerSecond

Implements AudioQualityInterface.

6.138.3.6 card8 SpectrumAnalyzer::getChannels ( ) const [virtual]

getChannels() Implementation of AudioQualityInterface.

See also

AudioQualityInterface::getChannels

Implements AudioQualityInterface.

6.138.3.7 card16 SpectrumAnalyzer::getSamplingRate ( ) const [virtual]

getSamplingRate() Implementation of AudioQualityInterface.

See also

AudioQualityInterface::getSamplingRate

Implements AudioQualityInterface.

6.138.3.8 bool SpectrumAnalyzer::getSpectrum (cardinal * *left*, cardinal * *right*, const cardinal *bars*)

Do Fourier transformation and get spectrum.

Parameters

left	Pointer to spectrum array for left channel.
right	Pointer to spectrum array for right channel.
bars	Number of bars.

Returns

true, if spectrum has been computed; false, if there is not enough input data available.

6.138.3.9 bool SpectrumAnalyzer::ready ( ) const [virtual]

ready() implementation of AudioWriterInterface

See also

AudioWriterInterface::ready

Implements AudioWriterInterface.

6.138.3.10 card8 SpectrumAnalyzer::setBits (const card8 *bits*) [virtual]

setBits() Implementation of AdjustableAudioQualityInterface.

See also

AdjustableAudioQualityInterface::setBits

Implements AdjustableAudioQualityInterface.

6.138.3.11 card16 SpectrumAnalyzer::setByteOrder (const card16 *byteOrder*) [virtual]

setByteOrder() Implementation of AdjustableAudioQualityInterface.

See also

AdjustableAudioQualityInterface::setByteOrder

Implements AdjustableAudioQualityInterface.

6.138.3.12 card8 SpectrumAnalyzer::setChannels (const card8 *channels*) [virtual]

setChannels() Implementation of AdjustableAudioQualityInterface.

See also

AdjustableAudioQualityInterface::setChannels

Implements AdjustableAudioQualityInterface.

6.138.3.13 card16 SpectrumAnalyzer::setSamplingRate (const card16 *samplingRate*) [virtual]

setSamplingRate() Implementation of AdjustableAudioQualityInterface.

See also

AdjustableAudioQualityInterface::setSamplingRate

Implements AdjustableAudioQualityInterface.

6.138.3.14 void SpectrumAnalyzer::sync ( ) [virtual]

sync() implementation of AudioWriterInterface

See also

AudioWriterInterface::sync

Implements AudioWriterInterface.

6.138.3.15 bool SpectrumAnalyzer::write (const void * *data*, const size_t *length*) [virtual]

write() implementation of AudioWriterInterface

See also

AudioWriterInterface::write

Implements AudioWriterInterface.

6.138.4 Member Data Documentation

6.138.4.1 card8 SpectrumAnalyzer::AudioBits [private]

6.138.4.2 card16 SpectrumAnalyzer::AudioByteOrder [private]

6.138.4.3 card8 SpectrumAnalyzer::AudioChannels [private]

6.138.4.4 card16 SpectrumAnalyzer::AudioSamplingRate [private]

6.138.4.5 FastFourierTransformation* SpectrumAnalyzer::FFT [private]

6.138.4.6 const cardinal SpectrumAnalyzer::FFTPoints = 256 [static, private]

6.138.4.7 char SpectrumAnalyzer::InputBuffer[4 *FFTPoints] [private]

6.138.4.8 cardinal SpectrumAnalyzer::InputBufferPos [private]

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

- spectrumanalyzer.h
- spectrumanalyzer.cc

6.139 StreamDescription Class Reference

Stream Description.

`#include <streamdescription.h>`

Public Member Functions

- StreamDescription ()
- ~StreamDescription ()
- void init (ManagedStreamInterface *stream, const ServiceLevelAgreement *sla, const cardinal maxPoints, const card64 bwThreshold, const double utThreshold, const double systemDelayTolerance, const bool unlayeredAllocation)
- bool tryAllocation (const ServiceLevelAgreement *sla, card64 &totalAvailable-Bandwidth, card64 *classAvailableBandwidthArray, ResourceUtilizationPoint &rup, const card64 bandwidthLimit=(card64)-1)

Public Attributes

- ManagedStreamInterface * Interface
- AbstractQoSDescription * QoSDescription
- SessionDescription * Session
- card64 StreamID
- cardinal Layers
- cardinal RUEntries
- ResourceUtilizationPoint RUList [MaxRUEntries]
- cardinal NewLayerClassNumber [RTPConstants::RTPMaxQualityLayers]
- card64 NewLayerClassBandwidth [RTPConstants::RTPMaxQualityLayers]
- double NewCostPerSecond
- ResourceUtilizationPoint NewQuality
- double LastUtilization
- cardinal CurrentLayerClassNumber [RTPConstants::RTPMaxQualityLayers]
- card64 CurrentLayerClassBandwidth [RTPConstants::RTPMaxQualityLayers]
- double CurrentCostPerSecond
- ResourceUtilizationPoint CurrentQuality
- card64 ReservationTimeStamp
- double TotalCost
- double TotalBandwidthUsage
- double TotalUtilization
- double TotalRuntime
- cardinal TotalReservationUpdates
- cardinal PartialRemappings
- cardinal CompleteRemappings
- cardinal Inits
- cardinal BufferFlushes
- card64 LastInitDuration
- double ReportedLossRate [RTPConstants::RTPMaxQualityLayers]
- double ReportedJitter [RTPConstants::RTPMaxQualityLayers]
- double MeasuredTransferDelay [TrafficClassValues::MaxValues]
- InternetAddress RoundTripTimeDestination
- card64 NextInterval
- bool MaximumReached
- bool UnlayeredAllocation

Private Member Functions

- bool calculatePossibleLayerClassMappings (ResourceUtilizationPoint &rup, const ServiceLevelAgreement *sla, const AbstractQoSDescription *aqd)

6.139.1 Detailed Description

Stream Description.

This class contains a description of a stream. It is used for the bandwidth manager's remapping algorithm.

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

6.139.2 Constructor & Destructor Documentation

6.139.2.1 StreamDescription::StreamDescription ( )

Constructor.

6.139.2.2 StreamDescription::~StreamDescription ( )

Destructor.

6.139.3 Member Function Documentation

6.139.3.1 bool StreamDescription::calculatePossibleLayerClassMappings (ResourceUtilizationPoint & *rup,* const ServiceLevelAgreement * *sla,* const AbstractQoSDescription * *aqd*) [private]

6.139.3.2 void StreamDescription::init (ManagedStreamInterface * *stream,* const ServiceLevelAgreement * *sla,* const cardinal *maxPoints,* const card64 *bwThreshold,* const double *utThreshold,* const double *systemDelayTolerance,* const bool *unlayeredAllocation*)

Initialize.

Parameters

stream	ManagedStreamInterface.
sla	Service level agreement.
maxPoints	Maximum number of resource/utilization points for each stream.
bwThreshold	Bandwidth threshold.
utThreshold	Utilization threshold.
system-Delay-Tolerance	The system's delay tolerance for the buffering optimization.
unlayered-Allocation	True, to use the same DiffServ class for ∗all∗ layers; false otherwise.

6.139.3.3 bool StreamDescription::tryAllocation (const ServiceLevelAgreement ∗ *sla,* card64 & *totalAvailableBandwidth,* card64 ∗ *classAvailableBandwidthArray,* ResourceUtilizationPoint & *rup,* const card64 *bandwidthLimit =* (card64) −1)

Try to allocate given layer bandwidths to a stream. If allocation is successful, the availability references are decremented by the bandwidth allocation. The allocated bandwidths and buffer delays will be stored into the resource/utilization point.

Parameters

sla	Service level agreement.
total-Available-Bandwidth	Reference to total available bandwidth.
class-Available-Bandwidth-Array	Available bandwidths for each DiffServ class.
rup	Resource/utilization point to do allocation for.
session	Session of the stream.
bandwidth-Limit	Upper bandwidth limit (e.g. the session's maximum bandwidth).

6.139.4 Member Data Documentation

6.139.4.1 cardinal StreamDescription::BufferFlushes

Number of buffer flushes.

6.139.4.2 cardinal StreamDescription::CompleteRemappings

Number of complete remappings forced by this stream.

6.139.4.3 double StreamDescription::CurrentCostPerSecond

Current bandwidth cost per second.

6.139.4.4 card64 StreamDescription::CurrentLayerClassBandwidth[RTPConstants::RTPMaxQualityLayers]

Current layer's allocated bandwidth currently used.

6.139.4.5 cardinal StreamDescription::CurrentLayerClassNumber[RTPConstants::RTPMaxQualityLayers]

Layer's allocated DiffServ class number currently used.

6.139.4.6 ResourceUtilizationPoint StreamDescription::CurrentQuality

Current quality.

6.139.4.7 cardinal StreamDescription::Inits

Number of init() calls.

See also

init

6.139.4.8 ManagedStreamInterface* StreamDescription::Interface

Stream's manager interface.

6.139.4.9 card64 StreamDescription::LastInitDuration

Duration of last init() call.

6.139.4.10 double StreamDescription::LastUtilization

Last *complete* remapping's utilization.

6.139.4.11 cardinal StreamDescription::Layers

Number of layers.

6.139.4.12 bool StreamDescription::MaximumReached

True, if all following higher bandwidth allocations will fail (no more bandwidth available to achieve higher quality -> no more allocation trials necessary); false otherwise.

6.139.4.13 double StreamDescription::MeasuredTransferDelay[TrafficClassValues::MaxValues]

Smoothed measured transfer delay from ICMP echo replies for each DiffServ class.

6.139.4.14 double StreamDescription::NewCostPerSecond

New remapping's bandwidth cost per second.

6.139.4.15 card64 StreamDescription::NewLayerClassBandwidth[RTPConstants::RTPMaxQualityLayers]

New remapping's layer's allocated bandwidth.

6.139.4.16 cardinal StreamDescription::NewLayerClassNumber[RTPConstants::RTPMaxQualityLayers]

New remapping's layer's allocated DiffServ class number.

6.139.4.17 ResourceUtilizationPoint StreamDescription::NewQuality

New remapping's quality.

6.139.4.18 card64 StreamDescription::NextInterval

Time to next interval in microseconds.

6.139.4.19 cardinal StreamDescription::PartialRemappings

Number of successful partial remappings.

6.139.4.20 AbstractQoSDescription* StreamDescription::QoSDescription

Stream's AbstractQoSDescription.

6.139.4.21 double StreamDescription::ReportedJitter[RTPConstants::RTPMaxQualityLayers]

Smoothed reported jitter for each layer (via RTCP reports).

6.139.4.22 double StreamDescription::ReportedLossRate[RTPConstants::RTPMaxQualityLayers]

Smoothed received loss rate for each layer (via RTCP reports).

6.139.4.23 card64 StreamDescription::ReservationTimeStamp

Reservation time stamp.

6.139.4.24 InternetAddress StreamDescription::RoundTripTimeDestination

Round trip time measurement destination.

6.139.4.25 cardinal StreamDescription::RUEntries

Number of entries in resource/utilization list.

6.139.4.26 ResourceUtilizationPoint StreamDescription::RUList[MaxRUEntries]

Resource/utilization list.

6.139.4.27 SessionDescription* StreamDescription::Session

Session description.

6.139.4.28 card64 StreamDescription::StreamID

Stream ID.

6.139.4.29 double StreamDescription::TotalBandwidthUsage

Total bandwidth usage of this stream.

6.139.4.30 double StreamDescription::TotalCost

Total cost of this stream.

6.139.4.31 cardinal StreamDescription::TotalReservationUpdates

Total reservation updates.

6.139.4.32 double StreamDescription::TotalRuntime

Total runtime of this stream.

6.139.4.33 double StreamDescription::TotalUtilization

Total utilization of this stream.

6.139.4.34 bool StreamDescription::UnlayeredAllocation

Unlayered allocation: Use the same DiffServ class for ∗all∗ layers.

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

- streamdescription.h
- streamdescription.cc

6.140 String Class Reference

String.

```
#include <tdstrings.h>
```

Public Member Functions

- String ()
- String (const String &string)
- String (const char ∗string)
- String (const char ∗string, const cardinal length)
- String (const cardinal value)
- ∼String ()
- const char ∗ getData () const
- cardinal length () const
- bool isNull () const
- integer index (const char c) const
- integer rindex (const char c) const
- integer find (const String &string) const
- String toUpper () const
- String toLower () const
- String left (const cardinal maxChars) const

- String mid (const cardinal start, const cardinal maxChars) const
- String mid (const cardinal start) const
- String right (const cardinal maxChars) const
- String stripWhiteSpace () const
- bool scanSetting (String &s1, String &s2) const
- String & operator= (const String &string)
- String & operator= (const char *string)
- String & operator= (const cardinal value)
- int operator== (const String &string) const
- int operator!= (const String &string) const
- int operator< (const String &string) const
- int operator<= (const String &string) const
- int operator> (const String &string) const
- int operator>= (const String &string) const
- char operator[] (const int index) const

Static Public Member Functions

- static cardinal stringLength (const char *string)
- static integer stringCompare (const char *str1, const char *str2)
- static char * stringDuplicate (const char *string)

Private Member Functions

- void setData (char *string)

Private Attributes

- char * Data

6.140.1 Detailed Description

String.

This class implements the String datatype.

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

6.140.2 Constructor & Destructor Documentation

6.140.2.1 String::String ( )

Constructor for an empty string.

6.140.2.2 String::String (const String & *string*)

Constructor for a copy of a string.

Parameters

string	String to be copied.

6.140.2.3 String::String (const char * *string*)

Constructor for a copy of a string.

Parameters

string	String to be copied.

6.140.2.4 String::String (const char * *string*, const cardinal *length*)

Constructor for a copy of a string with a given length to be copied.

Parameters

string	String to be copied.
length	Number of bytes to be copied.

6.140.2.5 String::String (const cardinal *value*)

Constructor for a string from a number.

Parameters

value	Number.

6.140.2.6 String::~String ( )

Destructor.

6.140.3 Member Function Documentation

6.140.3.1 integer String::find (const String & *string*) const [inline]

Find first position of a string in a string

Parameters

string	String to find in string.

Returns

Position of -1, if string is not in string.

6.140.3.2 const char* String::getData ( ) const [inline]

Get string data.

Returns

String data.

6.140.3.3 integer String::index (const char *c*) const [inline]

Find first position of a character in string.

Parameters

c	Character.

Returns

Position of -1, if character is not in string.

6.140.3.4 bool String::isNull ( ) const [inline]

Check, if string is NULL.

Returns

true, if string is NULL; false otherwise.

6.140.3.5 String String::left (const cardinal *maxChars*) const

Get left part of string.

Parameters

maxChars	Maximum number of characters to be copied.

Returns

String.

6.140.3.6 cardinal String::length ( ) const [inline]

Get string length.

Returns

Length in bytes.

6.140.3.7 String String::mid (const cardinal *start*, const cardinal *maxChars*) const

Get middle part of string.

Parameters

start	Start position in String.
maxChars	Maximum number of characters to be copied.

Returns

String.

6.140.3.8 String String::mid (const cardinal *start*) const [inline]

Get part from start to end of string.

Parameters

start	Start position in String.

Returns

String.

6.140.3.9 int String::operator!= (const String & *string*) const [inline]

Implementation of != operator.

6.140.3.10 int String::operator< (const String & *string*) const [inline]

Implementation of < operator.

6.140.3.11 int String::operator<= (const String & *string*) const [inline]

Implementation of <= operator.

6.140.3.12 String & String::operator= (const String & *string*)

Implementation of = operator.

6.140.3.13 String & String::operator= (const char * *string*)

Implementation of = operator.

6.140.3.14 String & String::operator= (const cardinal *value*)

Implementation of = operator.

6.140.3.15 int String::operator== (const String & *string*) const [inline]

Implementation of == operator.

6.140.3.16 int String::operator> (const String & *string*) const [inline]

Implementation of > operator.

6.140.3.17 int String::operator>= (const String & *string*) const [inline]

Implementation of >= operator.

6.140.3.18 char String::operator[] (const int *index*) const [inline]

Implementation of [] operator.

6.140.3.19 String String::right (const cardinal *maxChars*) const

Get right part of string.

Parameters

maxChars	Maximum number of characters to be copied.

Returns

String.

6.140.3.20 integer String::rindex (const char *c*) const [inline]

Find last position of a character in string.

Parameters

c	Character.

Returns

Position of -1, if character is not in string.

6.140.3.21 bool String::scanSetting (String & *s1*, String & *s2*) const

Scan setting string, e.g. " FileName = Test.file ". Spaces are removed, the first string (name) is converted to uppercase. The second string (value) may contain "-chars for values with spaces. The "-chars will be removed from the result.

Parameters

name	Reference to store the name.
value	Reference to store the value.

Returns

true, if scan was successful; false otherwise.

6.140.3.22 void String::setData (char * *string*) [inline, private]

6.140.3.23 static integer String::stringCompare (const char * *str1*, const char * *str2*) [inline, static]

Compare two strings.

Parameters

str1	First string.
str2	Second string.

Returns

str1 < str1 => -1; str1 == str2 => 0; str1 > str2 => 1.

6.140.3.24 static char* String::stringDuplicate (const char * *string*) [inline, static]

Duplicate a string. The new string can be deallocated with the delete operator.

Parameters

string	String to be duplicated.

Returns

New string.

6.140.3.25 static cardinal String::stringLength (const char * *string*) [inline, static]

Compute length of a string.

Parameters

string	String.

Returns

Length.

6.140.3.26 String String::stripWhiteSpace ( ) const

Get string with spaces from beginning and end of the string removed.

Returns

New string.

6.140.3.27 String String::toLower ( ) const

Get lowercase string from string.

Returns

Lowercase string.

6.140.3.28 String String::toUpper ( ) const

Get uppercase string from string.

Returns

Uppercase string.

6.140.4 Member Data Documentation

6.140.4.1 char* String::Data [private]

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

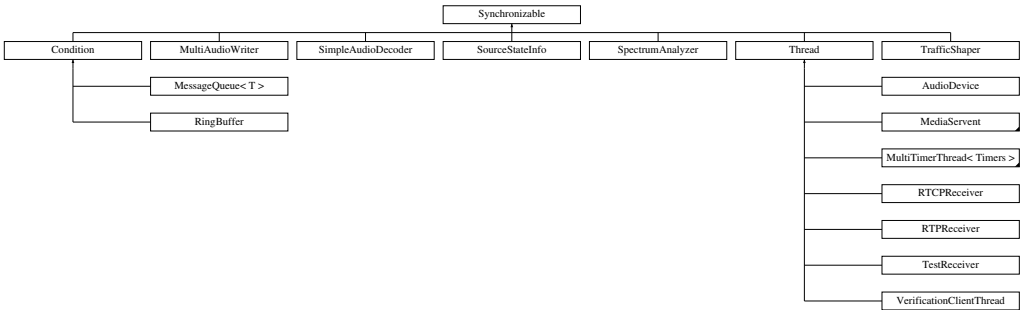
- tdstrings.h
- tdstrings.cc

6.141 Synchronizable Class Reference

Synchronizable.

`#include <synchronizable.h>`

Inheritance diagram for Synchronizable:



Public Member Functions

- Synchronizable (const char *name="Synchronizable", const bool recursive=true)
- ~Synchronizable ()
- void synchronized ()
- bool synchronizedTry ()
- void unsynchronized ()
- void resynchronize ()
- const char * getName () const
- void setName (const char *name)

Static Public Member Functions

- static bool setCancelState (const bool enabled)

Protected Attributes

- pthread_mutex_t Mutex
- bool Recursive
- char MutexName [64]

6.141.1 Detailed Description

Synchronizable.

This class realizes synchronized access to a thread's data by other threads. - Synchronization is done by using a global pthread mutex and obtaining access to this mutex by synchronized() for synchronized access and releasing this mutex for unsynchronized access. IMPORTANT: Do ∗not∗ use synchronized()/unsynchronized() within async signal handlers. This may cause deadlocks. See PThread's pthread_mutex_lock man-page, section "Async Signal Safety" for more information!

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

See also

Thread

6.141.2 Constructor & Destructor Documentation

6.141.2.1 Synchronizable::Synchronizable (const char ∗ *name* = "Synchronizable", const bool *recursive* = true)

Constructor.

Parameters

name	Name.
recursive	true to make mutex recursive (default); false otherwise.

6.141.2.2 Synchronizable::~Synchronizable ( )

Destructor.

6.141.3 Member Function Documentation

6.141.3.1 const char* Synchronizable::getName ( ) const [inline]

Get name of synchronizable object.

Returns

Name.

6.141.3.2 void Synchronizable::resynchronize ( )

Do reinitialization of Synchronizable.

6.141.3.3 static bool Synchronizable::setCancelState (const bool *enabled*) [inline, static]

Enable or disable cancelability of calling thread.

Parameters

enabled	true to enable cancellation; false otherwise.

6.141.3.4 void Synchronizable::setName (const char * *name*) [inline]

Set name of synchronizable object.

Parameters

name	Name.

6.141.3.5 void Synchronizable::synchronized ( ) [inline]

synchronized() begins a synchronized block. The block has to be finished by unsynchronized(). synchronized() will wait until the mutex is available.

See also

unsynchronized
synchronizedTry

6.141.3.6 bool Synchronizable::synchronizedTry ( ) [inline]

synchronizedTry() tries to begins a synchronized block. It does the same as synchronized(), but returns immediately, if the mutex is obtained by another thread.

See also

synchronized
unsynchronized

6.141.3.7 void Synchronizable::unsynchronized ( ) [inline]

unsynchronized() ends a synchronized block, which has begun by synchronized().

See also

synchronized

6.141.4 Member Data Documentation

6.141.4.1 pthread_mutex_t Synchronizable::Mutex [protected]

6.141.4.2 char Synchronizable::MutexName[64] [protected]

6.141.4.3 bool Synchronizable::Recursive [protected]

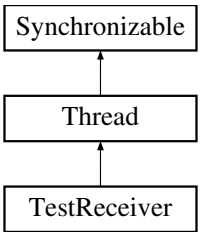
The documentation for this class was generated from the following files:

- synchronizable.h
- synchronizable.cc

6.142 TestReceiver Class Reference

RTCP Receiver.

Inheritance diagram for TestReceiver:



Public Member Functions

- TestReceiver ()
- TestReceiver (RTPAdaptionLayer ∗server, Socket ∗receiverSocket)
- ∼TestReceiver ()
- void init (RTPAdaptionLayer ∗server, Socket ∗receiverSocket)

Private Member Functions

- void run ()

Private Attributes

- Socket ∗ ReceiverSocket
- RTPAdaptionLayer ∗ Server
- double AverageRTCPSize

6.142.1 Detailed Description

RTCP Receiver.

This class implements an RTCP receiver based on Thread.

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

6.142.2 Constructor & Destructor Documentation

6.142.2.1 TestReceiver::TestReceiver ( )

Default constructor. You have to initialize TestReceiver by calling init(...) later!

See also

init

6.142.2.2 TestReceiver::TestReceiver (RTPAdaptionLayer ∗ *server*, Socket ∗ *receiverSocket*)

Constructor for new TestReceiver. The new receiver's thread has to be started by calling start()!

Parameters

server	RTPAdaptionLayer.
receiver-Socket	Socket to receive RTCP packets from.

6.142.2.3 TestReceiver::~TestReceiver ( )

Destructor.

6.142.3 Member Function Documentation

6.142.3.1 void TestReceiver::init (RTPAdaptionLayer * *server,* Socket * *receiverSocket*)

Initialize new TestReceiver. The new receiver's thread has to be started by calling start()!

Parameters

server	RTPAdaptionLayer.
receiver-Socket	Socket to receive RTCP packets from.

6.142.3.2 void TestReceiver::run ( ) [private, virtual]

The virtual run() method, which contains the thread's implementation. It has to be implemented by classes, which inherit Thread.

Implements Thread.

6.142.4 Member Data Documentation

6.142.4.1 double TestReceiver::AverageRTCPSize [private]

6.142.4.2 Socket* TestReceiver::ReceiverSocket [private]

6.142.4.3 RTPAdaptionLayer* TestReceiver::Server [private]

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

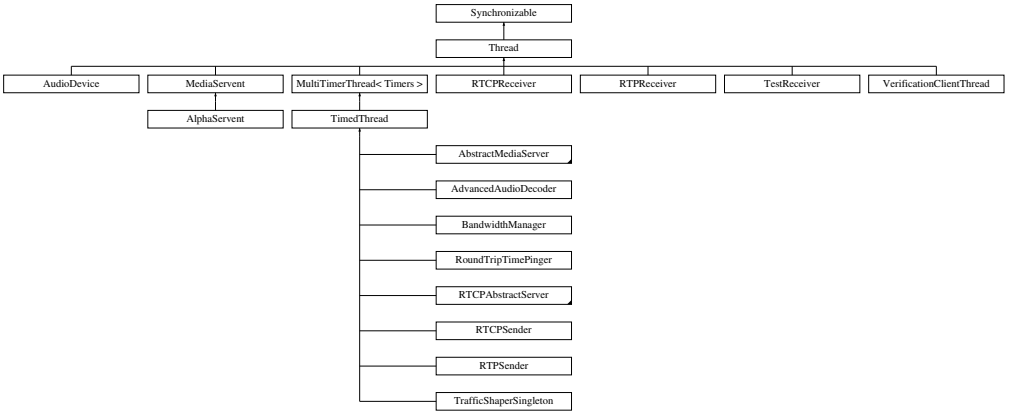
- s2.cc

6.143 Thread Class Reference

Thread.

`#include <thread.h>`

Inheritance diagram for Thread:



Public Member Functions

- Thread (const char *name="Thread", const cardinal flags=TF_CancelDeferred)
- virtual ~Thread ()
- bool running () const
- pid_t getPID () const
- virtual bool start (const char *name=NULL)
- virtual void * stop ()
- void * join ()
- virtual void cancel ()

Static Public Member Functions

- static card64 delay (const card64 delayTimeout, const bool interruptable=false)
- static cardinal setCancelState (const cardinal state)

Static Public Attributes

- static const cardinal TF_CancelAsynchronous = 0
- static const cardinal TF_CancelDeferred = (1 << 0)
- static const cardinal TCS_CancelEnabled = PTHREAD_CANCEL_ENABLE
- static const cardinal TCS_CancelDisabled = PTHREAD_CANCEL_DISABLE
- static const cardinal TCS_CancelDeferred = PTHREAD_CANCEL_DEFERRED

Protected Member Functions

- virtual void testCancel ()
- virtual void run ()=0

Static Protected Member Functions

- static void exit (void *result=NULL)
- static void yield ()

Protected Attributes

- pthread_t PThread
- pid_t PID

Static Private Member Functions

- static void * go (void *argument)

Private Attributes

- cardinal Flags
- pthread_mutex_t StartupMutex
- pthread_cond_t StartupCondition

6.143.1 Detailed Description

Thread.

This abstract class realizes threads based on Linux's pthreads. The user of this class has to implement run(). Synchronization is implemented by inheriting Synchronizable. IMPORTANT: Do *not* use Thread methods within async signal handlers. This may cause deadlocks. See PThread's pthread_mutex_lock man-page, section "Async Signal Safety" for more information!

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

See also

Synchronizable

6.143.2 Constructor & Destructor Documentation

6.143.2.1 Thread::Thread (const char * *name* = "Thread", const cardinal *flags* = TF_CancelDeferred)

Constructor. A new thread will be created but *not* started! To start the new thread, call start().

Parameters

name	Thread name.
flags	Flags for the thread to be created.

See also

start

6.143.2.2 Thread::~Thread ( ) [virtual]

Destructor. The thread will be stopped (if running) and deleted.

6.143.3 Member Function Documentation

6.143.3.1 void Thread::cancel ( ) [virtual]

Cancel the thread.

Reimplemented in MultiTimerThread< Timers >.

6.143.3.2 card64 Thread::delay (const card64 *delayTimeout*, const bool *interruptable* = false) [static]

Delay execution of current thread for a given timeout. This function uses nanosleep(), so no signals are affected.

Parameters

delayTime	Timeout in microseconds.
interruptable	true, if delay may be interrupted by signals; false otherwise.

Returns

Remaining delay, if interrupted; 0 otherwise.

6.143.3.3 static void Thread::exit (void * *result* = NULL) [inline, static, protected]

Exit current thread.

Parameters

result	Result to return.

6.143.3.4 pid_t Thread::getPID ( ) const [inline]

Get thread's PID.

6.143.3.5 void * Thread::go (void * *argument*) [static, private]

6.143.3.6 void * Thread::join ( )

Wait for the thread to be finished.

Returns

Return value from joined thread.

6.143.3.7 virtual void Thread::run ( ) [protected, pure virtual]

The virtual run() method, which contains the thread's implementation. It has to be implemented by classes, which inherit Thread.

Implemented in TestReceiver, MediaServent, MultiTimerThread< Timers >, AudioDevice, RTPReceiver, RTCPReceiver, and VerificationClientThread.

6.143.3.8 bool Thread::running ( ) const [inline]

Check, if the thread is running.

Returns

true, if the thread is running, false otherwise

6.143.3.9 static cardinal Thread::setCancelState (const cardinal *state*) [inline, static]

Enable or disable cancelability of calling thread. The previous state is returned. - Important note: The result may include additional state information, depending on the operating system. This state can be restored by giving this complete information to a setCancelState() call.

Parameters

enabled	TCS_CancelEnable to enable cancellation; TCS_CancelDisable otherwise.

Returns

Old state.

6.143.3.10 bool Thread::start (const char * *name* = NULL) [virtual]

Start the thread, if not already started.

Parameters

name	Thread name (NULL for default).

Returns

true, if the thread has been started; false, if not.

6.143.3.11 void * Thread::stop ( ) [virtual]

Stop the thread, if not already stopped. If the thread flag ThreadCancelAsynchronous is set, it will be stopped immediately. If the flag ThreadCancelDeferred is set, it will be stopped when a cancellation point is reached (-> see pthreads documentation). testCancel() is such a cancellation point.

See also

testCancel

Returns

Return value from stopped thread.

Reimplemented in AbstractMediaServer, MultiTimerThread< Timers >, RTCPAbstractServer, and VerificationClientThread.

6.143.3.12 void Thread::testCancel ( ) [protected, virtual]

Test for cancellation. If the thread received a cancel signal, it will be cancelled.

6.143.3.13 static void Thread::yield ( ) [inline, static, protected]

Voluntarily move current thread to end of queue of threads waiting for CPU time (sched_yield() call). This will result in scheduling to next waiting thread, if there is any.

6.143.4 Member Data Documentation

6.143.4.1 cardinal Thread::Flags [private]

6.143.4.2 pid_t Thread::PID [protected]

6.143.4.3 pthread_t Thread::PThread [protected]

6.143.4.4 pthread_cond_t Thread::StartupCondition [private]

6.143.4.5 pthread_mutex_t Thread::StartupMutex [private]

6.143.4.6 const cardinal Thread::TCS_CancelDeferred = PTHREAD_CANCEL_DEFERRED [static]

6.143.4.7 const cardinal Thread::TCS_CancelDisabled = PTHREAD_CANCEL_DISABLE [static]

6.143.4.8 const cardinal Thread::TCS_CancelEnabled = PTHREAD_CANCEL_ENABLE [static]

6.143.4.9 const cardinal Thread::TF_CancelAsynchronous = 0 [static]

6.143.4.10 const cardinal Thread::TF_CancelDeferred = (1 << 0) [static]

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

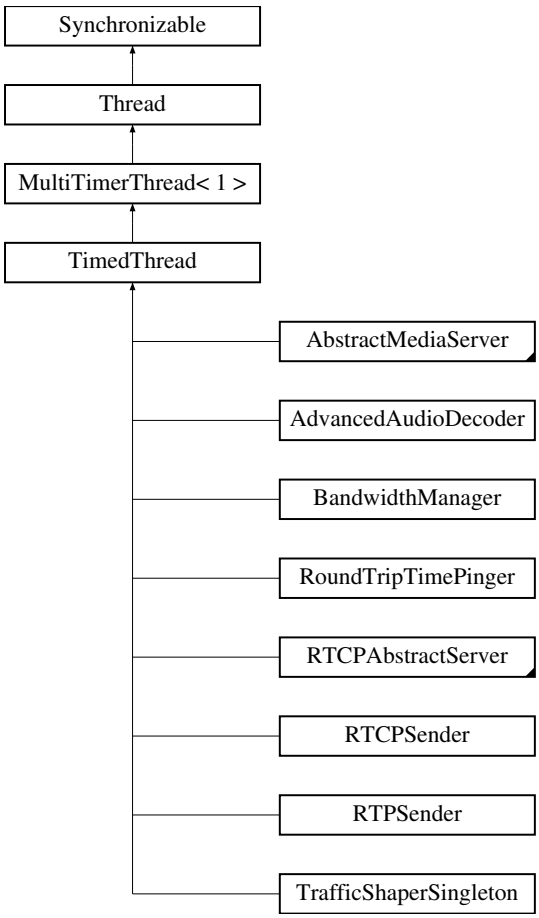
- thread.h
- thread.cc

6.144 TimedThread Class Reference

Timed Thread.

`#include <timedthread.h>`

Inheritance diagram for TimedThread:



Public Member Functions

- TimedThread (const card64 usec, const char *name="TimedThread", const cardinal flags=TF_CancelDeferred)
- ~TimedThread ()
- card64 getInterval ()
- void setInterval (const card64 usec)
- void setNextAction (const card64 usec=0, const card64 callLimit=1)
- void setNextActionAbs (const card64 timeStamp=0, const card64 callLimit=1)
- cardinal getTimerCorrection ()
- void setTimerCorrection (const cardinal maxCorrection=0)
- void leaveCorrectionLoop ()
- void setFastStart (const bool on)
- bool getFastStart () const

Protected Member Functions

- virtual void timerEvent ()=0

Private Member Functions

- void timerEvent (const cardinal timer)

6.144.1 Detailed Description

Timed Thread.

This abstract class realizes a timed thread based on MultiTimerThread. The user of this class has to implement timerEvent(). Inaccurate system timers are corrected by calling user's timerEvent() implementation multiple times if necessary. This feature can be modified by setTimerCorrection (Default is on at a maximum of 10 calls).

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

See also

Thread

6.144.2 Constructor & Destructor Documentation

6.144.2.1 TimedThread::TimedThread (const card64 *usec*, const char * *name* = "TimedThread", const cardinal *flags* = TF_CancelDeferred)

Constructor. A new timed thread with a given interval will be created but *not* started! To start the new thread, call start(). The interval gives the time for the interval in microseconds, the virtual function timerEvent() is called. The default timer correction is set to 10. See setTimerCorrection() for more information on timer correction. The first call of timerEvent() will be made immediately, if the fast start option is set (default). Otherwise it will be made after the given interval.

Parameters

usec	Interval in microseconds.
name	Thread name.
flags	Thread flags.

See also

Thread::start
timerEvent
Thread::Thread

setTimerCorrection
setFastStart

6.144.2.2 TimedThread::~TimedThread ( )

Destructor.

6.144.3 Member Function Documentation

6.144.3.1 bool TimedThread::getFastStart ( ) const [inline]

Get fast start option: If false, the first call of timerEvent() will be made *after* the given interval; otherwise it will be made immediately.

Returns

true, if option is set; false otherwise.

6.144.3.2 card64 TimedThread::getInterval ( ) [inline]

Get timed thread's interval.

Returns

Interval in microseconds.

6.144.3.3 cardinal TimedThread::getTimerCorrection ( ) [inline]

Get maxCorrection value for inaccurate system timer.

Returns

true, if activated; false if not.

See also

setTimerCorrection

6.144.3.4 void TimedThread::leaveCorrectionLoop ( ) [inline]

Leave timer correction loop: If the thread is in a timer correction loop, the loop will be finished after the current timerEvent() call returns.

6.144.3.5 void TimedThread::setFastStart (const bool *on*) [inline]

Set fast start option: If false, the first call of timerEvent() will be made ∗after∗ the given interval; otherwise it will be made immediately. The default is true.

Parameters

on	true, to set option; false otherwise.

6.144.3.6 void TimedThread::setInterval (const card64 *usec*) [inline]

Set timed thread's interval.

Parameters

usec	Interval in microseconds.

6.144.3.7 void TimedThread::setNextAction (const card64 *usec* = 0, const card64 *callLimit* = 1) [inline]

Like setInterval(), but disabling FastStart first. This method can be used e.g. for a single shot timer.

Parameters

usec	Time to next invokation (0 = immediately).
callLimit	Call count limit (0 for infinite, default: 1).

See also

setInterval

6.144.3.8 void TimedThread::setNextActionAbs (const card64 *timeStamp* = 0, const card64 *callLimit* = 1) [inline]

Like setNextAction(), but the time stamp of the next invokation is given as absolute time (microseconds since January 01, 1970).

Parameters

usec	Time to next invokation (0 = immediately).
callLimit	Call count limit (0 for infinite, default: 1).

See also

setInterval
setNextAction

6.144.3.9 void TimedThread::setTimerCorrection (const cardinal *maxCorrection* = 0) [inline]

Set correction of inaccurate system timer to given value. This on will cause the timerEvent() function to be called a maximum of maxCorrection times, if the total number of calls is lower than the calculated number of times the function should have been called. If the number of correction calls is higher than maxCorrection, ∗no∗ correction will be done! Default is 0, which turns correction off.

Parameters

of	true to activate correction; false to deactivate.

6.144.3.10 virtual void TimedThread::timerEvent ( ) [protected, pure virtual]

The virtual timerEvent() method, which contains the timed thread's implementation. It has to be implemented by classes, which inherit TimedThread. This method is called regularly with the given interval.

Implemented in AbstractMediaServer, BandwidthManager, RTCPAbstractServer, RoundTripTimePinger, RTPSender, AdvancedAudioDecoder, RTCPSender, and TrafficShaperSingleton.

6.144.3.11 void TimedThread::timerEvent (const cardinal *timer*) [private, virtual]

The virtual timerEvent() method, which contains the multitimer thread's implementation. It has to be implemented by classes, which inherit MultiTimerThread. This method is called regularly with the given interval.

Implements MultiTimerThread< Timers >.

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

- timedthread.h
- timedthread.cc

6.145 MultiTimerThread< Timers >::TimerParameters Struct Reference

Public Attributes

- card64 Interval
- card64 CallLimit
- cardinal TimerCorrection
- bool FastStart
- bool Running
- bool Updated
- bool LeaveCorrectionLoop

template<const cardinal Timers> struct MultiTimerThread< Timers >::TimerParameters

6.145.1 Member Data Documentation

6.145.1.1 template<const cardinal Timers> card64 MultiTimerThread< Timers >::TimerParameters::CallLimit

6.145.1.2 template<const cardinal Timers> bool MultiTimerThread< Timers >::TimerParameters::FastStart

6.145.1.3 template<const cardinal Timers> card64 MultiTimerThread< Timers >::TimerParameters::Interval

6.145.1.4 template<const cardinal Timers> bool MultiTimerThread< Timers >::TimerParameters::LeaveCorrectionLoop

6.145.1.5 template<const cardinal Timers> bool MultiTimerThread< Timers >::TimerParameters::Running

6.145.1.6 template<const cardinal Timers> cardinal MultiTimerThread< Timers >::TimerParameters::TimerCorrection

6.145.1.7 template<const cardinal Timers> bool MultiTimerThread< Timers >::TimerParameters::Updated

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

- multitimerthread.h

6.146 TrafficClassValues Class Reference

Traffic Class Values.

`#include <trafficclassvalues.h>`

Static Public Member Functions

- static card8 getTrafficClassForIndex (const cardinal index)
- static const card16 getTrafficClassForName (const char *name)
- static const char * getNameForTrafficClass (const card8 trafficClass)
- static const char * getNameForIndex (const cardinal index)
- static cardinal getIndexForTrafficClass (const card8 trafficClass)

Static Public Attributes

- static const cardinal MaxValues = 16

Static Private Attributes

- static const card8 TCValues [MaxValues]
- static const char * TCNames [TrafficClassValues::MaxValues]

6.146.1 Detailed Description

Traffic Class Values.

This class contains a set of values for the traffic class/TOS byte of IP packets. This class contains only static methods and attributes.

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

6.146.2 Member Function Documentation

6.146.2.1 cardinal TrafficClassValues::getIndexForTrafficClass (const card8 *trafficClass*) [static]

Get index for given traffic class.

Parameters

trafficClass	Traffic class.

Returns

Index.

6.146.2.2 static const char* TrafficClassValues::getNameForIndex (const cardinal *index*) [inline, static]

Get name for index entry.

Parameters

index	Index.

Returns

Name.

6.146.2.3 const char * TrafficClassValues::getNameForTrafficClass (const card8 *trafficClass*) [static]

Get name for given traffic class.

Parameters

trafficClass	Traffic class.

Returns

Name.

6.146.2.4 static card8 TrafficClassValues::getTrafficClassForIndex (const cardinal *index*) [inline, static]

Get traffic class of given index.

Parameters

index	Index.

Returns

Traffic class.

6.146.2.5 const card16 TrafficClassValues::getTrafficClassForName (const char * *name*) [static]

Get traffic class for name.

Parameters

name	Name.

Returns

Traffic class or 0xffff, if name in unknown.

6.146.3 Member Data Documentation

6.146.3.1 const cardinal TrafficClassValues::MaxValues = 16 [static]

Number of values.

6.146.3.2 const char * TrafficClassValues::TCNames [static, private]

Initial value:

```
{
  "EF",
  "AF11","AF12","AF13",
  "AF21","AF22","AF23",
  "AF31","AF32","AF33",
  "AF41","AF42","AF43",
  "TD1", "TD2",
  "BE"
}
```

6.146.3.3 const card8 TrafficClassValues::TCValues [static, private]

Initial value:

```
{
  0xb8,
  0x28, 0x30, 0x38,
  0x48, 0x50, 0x58,
  0x68, 0x70, 0x78,
  0x88, 0x90, 0x98,
  0xa0, 0xa8,
  0x00
}
```

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

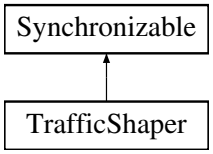
- trafficclassvalues.h
- trafficclassvalues.cc

6.147 TrafficShaper Class Reference

Traffic Shaper.

`#include <trafficshaper.h>`

Inheritance diagram for TrafficShaper:



Classes

- struct TrafficShaperPacket

Public Member Functions

- TrafficShaper ()
- TrafficShaper (Socket ∗socket)
- ∼TrafficShaper ()
- void init (Socket ∗socket)
- void setSocket (Socket ∗socket)
- card64 getBandwidth () const
- void setBandwidth (const card64 bandwidth)
- double getBufferDelay () const
- void setBufferDelay (const double bufferDelay)
- void flush ()
- bool refreshBuffer (const card8 trafficClass, const bool doRemapping)
- cardinal getLastSeqNum ()
- ssize_t sendTo (const void ∗buffer, const size_t length, const cardinal seqNum, const cardinal flags, const InternetFlow &receiver, const card8 trafficClass=0)
- ssize_t send (const void ∗buffer, const size_t length, const cardinal seqNum, const cardinal flags=0, const card8 trafficClass=0)
- ssize_t write (const void ∗buffer, const size_t length, const cardinal seqNum)

Private Types

- enum TrafficShaperCommand { TSC_Write = 0, TSC_Send = 1, TSC_SendTo = 2 }

Private Member Functions

- void sendAll ()
- ssize_t addPacket (const void ∗data, const cardinal bytes, const cardinal seqNum, InternetFlow &destination, const cardinal flags, const cardinal command)

Private Attributes

- std::deque< TrafficShaperPacket > Queue
- Socket * SenderSocket
- card64 SendTimeStamp
- card64 Bandwidth
- double BufferDelay
- integer LastError
- cardinal LastSeqNum

Static Private Attributes

- static TrafficShaperSingleton Singleton

Friends

- class TrafficShaperSingleton

6.147.1 Detailed Description

Traffic Shaper.

This class is a traffic shaper.

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

6.147.2 Member Enumeration Documentation

6.147.2.1 enum TrafficShaper::TrafficShaperCommand [private]

Enumerator:

TSC_Write

TSC_Send

TSC_SendTo

6.147.3 Constructor & Destructor Documentation

6.147.3.1 TrafficShaper::TrafficShaper ( )

Constructor.

6.147.3.2 TrafficShaper::TrafficShaper (Socket * *socket*)

Constructor.

6.147.3.3 TrafficShaper::~TrafficShaper ( )

Destructor.

6.147.4 Member Function Documentation

6.147.4.1 ssize_t TrafficShaper::addPacket (const void * *data,* const cardinal *bytes,* const cardinal *seqNum,* InternetFlow & *destination,* const cardinal *flags,* const cardinal *command*) [private]

6.147.4.2 void TrafficShaper::flush ( )

Flush buffer.

6.147.4.3 card64 TrafficShaper::getBandwidth ( ) const [inline]

Get bandwidth for following packets.

Returns

Bandwidth.

6.147.4.4 double TrafficShaper::getBufferDelay ( ) const [inline]

Get maximum buffer delay for following packets.

Returns

Maximum buffer delay in microseconds.

6.147.4.5 cardinal TrafficShaper::getLastSeqNum ( ) [inline]

Get sequence number of last packet sent.

Returns

Sequence number.

6.147.4.6 void TrafficShaper::init (Socket * *socket*)

Initialize.

Parameters

socket	Socket.

6.147.4.7 bool TrafficShaper::refreshBuffer (const card8 *trafficClass*, const bool *doRemapping*)

Adapt buffer's contents to changed bandwidth and delay settings.

Parameters

trafficClass	Traffic class to remap packets to.
doRemapping	true, to do traffic class remapping; false otherwise.

Returns

true, if buffer flush has been necessary; false otherwise.

6.147.4.8 ssize_t TrafficShaper::send (const void * *buffer*, const size_t *length*, const cardinal *seqNum*, const cardinal *flags* = 0, const card8 *trafficClass* = 0)

Wrapper for send(). send() will set the packet's traffic class, if trafficClass is not 0. In this case, the packet will be sent by sendto() to the destination address, the socket is connected to!

Parameters

buffer	Buffer with data to send.
length	Length of data to send.
seqNum	Packet's sequence number (-1 for none).
flags	Flags for sendto().
trafficClass	Traffic class for packet.

Returns

Bytes sent or error code < 0.

6.147.4.9 void TrafficShaper::sendAll ( ) [private]

6.147.4.10 ssize_t TrafficShaper::sendTo (const void * *buffer*, const size_t *length*, const cardinal *seqNum*, const cardinal *flags*, const InternetFlow & *receiver*, const card8 *trafficClass* = 0)

Wrapper for sendto(). sendto() will set the packet's traffic class, if trafficClass is not 0.

Parameters

buffer	Buffer with data to send.
length	Length of data to send.
seqNum	Packet's sequence number (-1 for none).
flags	Flags for sendto().
receiver	Address of receiver.

Returns

Bytes sent or error code < 0.

6.147.4.11 void TrafficShaper::setBandwidth (const card64 *bandwidth*) [inline]

Set bandwidth for following packets.

Parameters

bandwidth	Bandwidth.

6.147.4.12 void TrafficShaper::setBufferDelay (const double *bufferDelay*) [inline]

Set maximum buffer delay for following packets.

Parameters

bufferDelay	Maximum buffer delay in microseconds.

6.147.4.13 void TrafficShaper::setSocket (Socket * *socket*) [inline]

Set socket to send shaped traffic to.

Parameters

socket	Socket.

6.147.4.14 ssize_t TrafficShaper::write (const void * *buffer*, const size_t *length*, const cardinal *seqNum*)

Wrapper for write().

Parameters

buffer	Buffer with data to write
length	Length of data to write
seqNum	Packet's sequence number (-1 for none).

Returns

Bytes sent or error code < 0.

6.147.5 Friends And Related Function Documentation

6.147.5.1 friend class TrafficShaperSingleton [friend]

6.147.6 Member Data Documentation

6.147.6.1 card64 TrafficShaper::Bandwidth [private]

6.147.6.2 double TrafficShaper::BufferDelay [private]

6.147.6.3 integer TrafficShaper::LastError [private]

6.147.6.4 cardinal TrafficShaper::LastSeqNum [private]

6.147.6.5 std::deque<TrafficShaperPacket> TrafficShaper::Queue [private]

6.147.6.6 Socket* TrafficShaper::SenderSocket [private]

6.147.6.7 card64 TrafficShaper::SendTimeStamp [private]

6.147.6.8 TrafficShaperSingleton TrafficShaper::Singleton [static, private]

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

- trafficshaper.h
- trafficshaper.cc

6.148 TrafficShaper::TrafficShaperPacket Struct Reference

Public Member Functions

- int operator< (const TrafficShaperPacket &packet) const

Public Attributes

- card64 SendTimeStamp
- cardinal HeaderSize
- cardinal PayloadSize
- cardinal Flags
- cardinal Command
- InternetFlow Destination
- char * Data
- cardinal SeqNum

6.148.1 Member Function Documentation

6.148.1.1 int TrafficShaper::TrafficShaperPacket::operator< (const TrafficShaperPacket & *packet*) const [inline]

6.148.2 Member Data Documentation

6.148.2.1 cardinal TrafficShaper::TrafficShaperPacket::Command

6.148.2.2 char* TrafficShaper::TrafficShaperPacket::Data

6.148.2.3 InternetFlow TrafficShaper::TrafficShaperPacket::Destination

6.148.2.4 cardinal TrafficShaper::TrafficShaperPacket::Flags

6.148.2.5 cardinal TrafficShaper::TrafficShaperPacket::HeaderSize

6.148.2.6 cardinal TrafficShaper::TrafficShaperPacket::PayloadSize

6.148.2.7 card64 TrafficShaper::TrafficShaperPacket::SendTimeStamp

6.148.2.8 cardinal TrafficShaper::TrafficShaperPacket::SeqNum

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

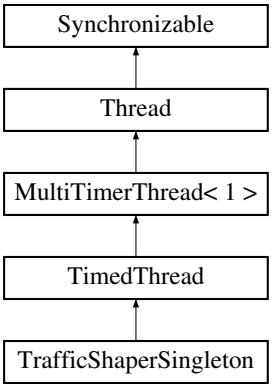
- trafficshaper.h

6.149 TrafficShaperSingleton Class Reference

Traffic Shaper Singleton.

`#include <trafficshaper.h>`

Inheritance diagram for TrafficShaperSingleton:



Public Member Functions

- TrafficShaperSingleton ()
- ~TrafficShaperSingleton ()
- void addTrafficShaper (TrafficShaper *ts)
- void removeTrafficShaper (TrafficShaper *ts)

Private Member Functions

- void timerEvent ()

Private Attributes

- std::vector< TrafficShaper * > ShaperSet
- cardinal UserCount

6.149.1 Detailed Description

Traffic Shaper Singleton.

This class is a singleton for the traffic shaper.

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

6.149.2 Constructor & Destructor Documentation

6.149.2.1 TrafficShaperSingleton::TrafficShaperSingleton ( )

Constructor.

6.149.2.2 TrafficShaperSingleton::~TrafficShaperSingleton ( )

Destructor.

6.149.3 Member Function Documentation

6.149.3.1 void TrafficShaperSingleton::addTrafficShaper (TrafficShaper * *ts*)

Add traffic shaper object.

Parameters

ts	TrafficShaper.

6.149.3.2 void TrafficShaperSingleton::removeTrafficShaper (TrafficShaper * *ts*)

Remove traffic shaper object.

Parameters

ts	TrafficShaper.

6.149.3.3 void TrafficShaperSingleton::timerEvent ( ) [private, virtual]

The virtual timerEvent() method, which contains the timed thread's implementation. It has to be implemented by classes, which inherit TimedThread. This method is called regularly with the given interval.

Implements TimedThread.

6.149.4 Member Data Documentation

6.149.4.1 std::vector<TrafficShaper*> TrafficShaperSingleton::ShaperSet [private]

6.149.4.2 cardinal TrafficShaperSingleton::UserCount [private]

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

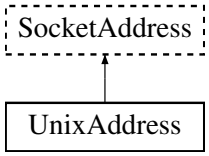
- trafficshaper.h
- trafficshaper.cc

6.150 UnixAddress Class Reference

Socket Address.

`#include <unixaddress.h>`

Inheritance diagram for UnixAddress:



Public Member Functions

- UnixAddress ()
- UnixAddress (const UnixAddress &address)
- UnixAddress (const String &name)
- UnixAddress (const sockaddr *address, const cardinal length)
- ~UnixAddress ()
- void reset ()
- SocketAddress * duplicate () const
- void init (const UnixAddress &address)
- void init (const String &name)
- UnixAddress & operator= (const UnixAddress &source)
- bool isValid () const
- integer getFamily () const
- String getAddressString (const cardinal format=PF_Default) const
- bool isNull () const
- card16 getPort () const
- void setPort (const card16 port)
- cardinal getSystemAddress (sockaddr *buffer, const socklen_t length, const cardinal type) const
- bool setSystemAddress (const sockaddr *address, const socklen_t length)
- int operator== (const UnixAddress &address) const
- int operator!= (const UnixAddress &address) const
- int operator< (const UnixAddress &address) const
- int operator<= (const UnixAddress &address) const
- int operator> (const UnixAddress &address) const
- int operator>= (const UnixAddress &address) const

Private Attributes

- char Name [MaxNameLength+1]

Static Private Attributes

- static const cardinal MaxNameLength = 108 - 1

6.150.1 Detailed Description

Socket Address.

This class manages an unix socket address.

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

6.150.2 Constructor & Destructor Documentation

6.150.2.1 UnixAddress::UnixAddress ( )

Constructor for an empty unix address.

6.150.2.2 UnixAddress::UnixAddress (const UnixAddress & *address*)

Constructor for an unix address from an unix address.

Parameters

address	Unix address.

6.150.2.3 UnixAddress::UnixAddress (const String & *name*)

Constructor for a unix address given by a string. Examples: "/tmp/test.socket".

Parameters

name	Address string.

6.150.2.4 UnixAddress::UnixAddress (const sockaddr * *address*, const cardinal *length*)

Constructor for a unix address from the system's sockaddr structure.

Parameters

address	sockaddr.
length	Length of sockaddr.

6.150.2.5 UnixAddress::~UnixAddress ( )

Destructor.

6.150.3 Member Function Documentation

6.150.3.1 SocketAddress * UnixAddress::duplicate ( ) const [virtual]

duplicate() implementation of SocketAddress.

See also

SocketAddress::duplicate

Implements SocketAddress.

6.150.3.2 String UnixAddress::getAddressString (const cardinal *format* = PF_Default) const [virtual]

getAddressString() implementation of SocketAddress.

See also

SocketAddress::getAddress

Implements SocketAddress.

6.150.3.3 integer UnixAddress::getFamily ( ) const [virtual]

getFamily() implementation of SocketAddress.

See also

SocketAddress::getFamily

Implements SocketAddress.

6.150.3.4 card16 UnixAddress::getPort ( ) const [virtual]

getPort() implementation of SocketAddress.

See also

SocketAddress::getPort

Implements SocketAddress.

6.150.3.5 cardinal UnixAddress::getSystemAddress (sockaddr * *buffer*, const socklen_t *length*, const cardinal *type*) const [virtual]

getSystemAddress() implementation of SocketAddress

See also

SocketAddress::getSystemAddress

Implements SocketAddress.

6.150.3.6 void UnixAddress::init (const UnixAddress & *address*)

Initialize unix address from unix address.

6.150.3.7 void UnixAddress::init (const String & *name*)

Initialize unix address from socket name.

6.150.3.8 bool UnixAddress::isNull ( ) const [inline]

Check, if the address is null.

Returns

true, if the address is not null; false otherwise.

6.150.3.9 bool UnixAddress::isValid ( ) const [virtual]

isValid() implementation of SocketAddress.

See also

SocketAddress::isValid

Implements SocketAddress.

6.150.3.10 int UnixAddress::operator!= (const UnixAddress & *address*) const [inline]

Implementation of != operator.

6.150.3.11 int UnixAddress::operator< (const UnixAddress & *address*) const

Implementation of < operator.

6.150.3.12 int UnixAddress::operator<= (const UnixAddress & *address*) const [inline]

Implementation of <= operator.

6.150.3.13 UnixAddress& UnixAddress::operator= (const UnixAddress & *source*) [inline]

Implementation of = operator.

6.150.3.14 int UnixAddress::operator== (const UnixAddress & *address*) const

Implementation of == operator.

6.150.3.15 int UnixAddress::operator> (const UnixAddress & *address*) const

Implementation of > operator.

6.150.3.16 int UnixAddress::operator>= (const UnixAddress & *address*) const [inline]

Implementation of >= operator.

6.150.3.17 void UnixAddress::reset ( ) [virtual]

Reset unix address.

Implements SocketAddress.

6.150.3.18 void UnixAddress::setPort (const card16 *port*) [virtual]

setPort() implementation of SocketAddress.

See also

SocketAddress::setPort

Implements SocketAddress.

6.150.3.19 bool UnixAddress::setSystemAddress (const sockaddr * *address*, const socklen_t *length*) [virtual]

setSystemAddress() implementation of SocketAddress.

See also

SocketAddress::setSystemAddress

Implements SocketAddress.

6.150.4 Member Data Documentation

6.150.4.1 const cardinal UnixAddress::MaxNameLength = 108 - 1 [static, private]

6.150.4.2 char UnixAddress::Name[MaxNameLength+1] [private]

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

- unixaddress.h
- unixaddress.cc

6.151 AudioServer::User Struct Reference

Public Attributes

- RTCPAbstractServer::Client * Client
- RTPSender Sender
- Socket SenderSocket
- InternetFlow Flow
- AudioEncoderRepository Repository
- MultiAudioReader Reader
- String MediaName
- integer StreamIdentifier
- card64 BandwidthLimit
- card16 LastSequenceNumber
- card16 PosChgSeqNumber
- bool UserLimitPause
- bool ManagerLimitPause
- bool ClientPause

6.151.1 Member Data Documentation

6.151.1.1 card64 AudioServer::User::BandwidthLimit

6.151.1.2 RTCPAbstractServer::Client* AudioServer::User::Client

6.151.1.3 bool AudioServer::User::ClientPause

6.151.1.4 InternetFlow AudioServer::User::Flow

6.151.1.5 card16 AudioServer::User::LastSequenceNumber

6.151.1.6 bool AudioServer::User::ManagerLimitPause

6.151.1.7 String AudioServer::User::MediaName

6.151.1.8 card16 AudioServer::User::PosChgSeqNumber

6.151.1.9 MultiAudioReader AudioServer::User::Reader

6.151.1.10 AudioEncoderRepository AudioServer::User::Repository

6.151.1.11 RTPSender AudioServer::User::Sender

6.151.1.12 Socket AudioServer::User::SenderSocket

6.151.1.13 integer AudioServer::User::StreamIdentifier

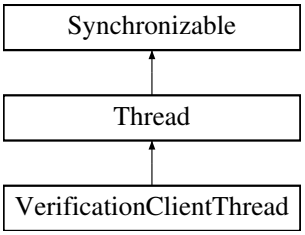
6.151.1.14 bool AudioServer::User::UserLimitPause

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

- audioserver.h

6.152 VerificationClientThread Class Reference

Inheritance diagram for VerificationClientThread:



Public Member Functions

- VerificationClientThread (const cardinal id, const char *localName, const char *url=DefaultURL, const cardinal count=DefaultMediaCount, const cardinal pause=DefaultPause, const double prSelectEncoding=0.1, const double prSelectQuality=0.7, const double prSelectPosition=0.3, const double prSelectMedia=0.05, const double prStop=0.03, const double prRestart=0.01)
- char * getStatusString (char *str, const cardinal maxLength)
- void * stop ()

Private Member Functions

- void run ()
- void test ()
- void writeLog (const char *str)
- void selectEncoding ()
- void selectQuality ()
- void selectPosition ()
- void selectMedia ()
- void restart ()

Private Attributes

- Randomizer Random
- AudioNull OutputDevice
- AudioClient * Client
- card32 SSRC
- card64 Position
- AudioQuality Quality
- const char * Encoding
- cardinal ID
- const char * URL
- cardinal MediaCount
- cardinal Pause
- double PrSelectEncoding
- double PrSelectQuality
- double PrSelectPosition
- double PrSelectMedia
- double PrStop
- double PrRestart
- String LocalName

6.152.1 Constructor & Destructor Documentation

6.152.1.1 VerificationClientThread::VerificationClientThread (const cardinal *id*, const char * *localName*, const char * *url* = DefaultURL, const cardinal *count* = DefaultMediaCount, const cardinal *pause* = DefaultPause, const double *prSelectEncoding* = 0.1, const double *prSelectQuality* = 0.7, const double *prSelectPosition* = 0.3, const double *prSelectMedia* = 0.05, const double *prStop* = 0.03, const double *prRestart* = 0.01)

6.152.2 Member Function Documentation

6.152.2.1 char * VerificationClientThread::getStatusString (char * *str*, const cardinal *maxLength*)

6.152.2.2 void VerificationClientThread::restart ( ) [private]

6.152.2.3 void VerificationClientThread::run ( ) [private, virtual]

The virtual run() method, which contains the thread's implementation. It has to be implemented by classes, which inherit Thread.

Implements Thread.

6.152.2.4 void VerificationClientThread::selectEncoding ( ) [private]

6.152.2.5 void VerificationClientThread::selectMedia ( ) [private]

6.152.2.6 void VerificationClientThread::selectPosition ( ) [private]

6.152.2.7 void VerificationClientThread::selectQuality ( ) [private]

6.152.2.8 void * VerificationClientThread::stop ( ) [virtual]

Stop the thread, if not already stopped. If the thread flag ThreadCancelAsynchronous is set, it will be stopped immediately. If the flag ThreadCancelDeferred is set, it will be stopped when a cancellation point is reached (-> see pthreads documentation). testCancel() is such a cancellation point.

See also

testCancel

Returns

Return value from stopped thread.

Reimplemented from Thread.

6.152.2.9 void VerificationClientThread::test ( ) [private]

6.152.2.10 void VerificationClientThread::writeLog (const char * *str*) [private]

6.152.3 Member Data Documentation

6.152.3.1 AudioClient* VerificationClientThread::Client [private]

6.152.3.2 const char* VerificationClientThread::Encoding [private]

6.152.3.3 cardinal VerificationClientThread::ID [private]

6.152.3.4 String VerificationClientThread::LocalName [private]

6.152.3.5 cardinal VerificationClientThread::MediaCount [private]

6.152.3.6 AudioNull VerificationClientThread::OutputDevice [private]

6.152.3.7 cardinal VerificationClientThread::Pause [private]

6.152.3.8 card64 VerificationClientThread::Position [private]

6.152.3.9 double VerificationClientThread::PrRestart [private]

6.152.3.10 double VerificationClientThread::PrSelectEncoding [private]

6.152.3.11 double VerificationClientThread::PrSelectMedia [private]

6.152.3.12 double VerificationClientThread::PrSelectPosition [private]

6.152.3.13 double VerificationClientThread::PrSelectQuality [private]

6.152.3.14 double VerificationClientThread::PrStop [private]

6.152.3.15 AudioQuality VerificationClientThread::Quality [private]

6.152.3.16 Randomizer VerificationClientThread::Random [private]

6.152.3.17 card32 VerificationClientThread::SSRC [private]

6.152.3.18 const char* VerificationClientThread::URL [private]

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

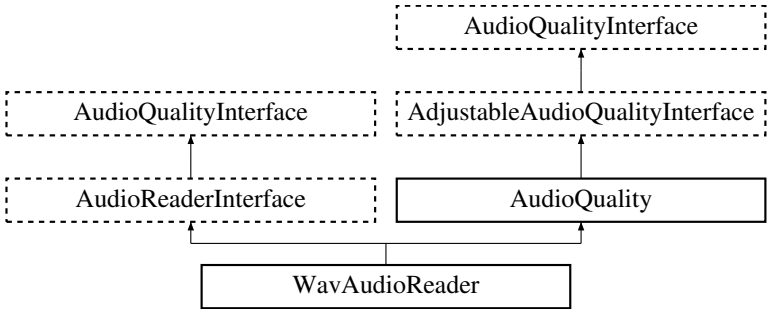
- rtpa-vclient.cc

6.153 WavAudioReader Class Reference

WAV Audio Reader.

`#include <wavaudioreader.h>`

Inheritance diagram for WavAudioReader:



Classes

- struct RIFF_Chunk
- struct RIFF_Header
- struct WAVE_Format

Public Member Functions

- WavAudioReader (const char *name=NULL)
- ~WavAudioReader ()
- bool openMedia (const char *name)
- void closeMedia ()
- bool ready () const
- void getMediaInfo (MediaInfo &mediaInfo) const
- MediaError getErrorCode () const
- card64 getPosition () const
- card64 getMaxPosition () const
- void setPosition (const card64 position)
- cardinal getNextBlock (void *buffer, const cardinal blockSize)

Private Member Functions

- bool getChunk (RIFF_Chunk &chunk)

Private Attributes

- MediaError Error
- FILE * InputFD
- WAVE_Format Format
- card64 StartPosition
- card64 EndPosition
- card64 Position
- card64 MaxPosition

6.153.1 Detailed Description

WAV Audio Reader.

This class is a reader for WAV audio files.

Author

Thomas Dreibholz (dreibh@iem.uni-due.de)

Version

1.0

6.153.2 Constructor & Destructor Documentation

6.153.2.1 WavAudioReader::WavAudioReader (const char * *name* = NULL)

Constructor.

Parameters

name	Name of WAV file or NULL.

6.153.2.2 WavAudioReader::~WavAudioReader ( )

Destructor.

6.153.3 Member Function Documentation

6.153.3.1 void WavAudioReader::closeMedia ( ) [virtual]

closeMedia() implementation of AudioReaderInterface.

See also

AudioReaderInterface::closeMedia

Implements AudioReaderInterface.

6.153.3.2 bool WavAudioReader::getChunk (RIFF_Chunk & *chunk*) [private]

6.153.3.3 MediaError WavAudioReader::getErrorCode ( ) const [virtual]

getErrorCode() implementation of AudioReaderInterface.

See also

AudioReaderInterface::getErrorCode

Implements AudioReaderInterface.

6.153.3.4 card64 WavAudioReader::getMaxPosition ( ) const [virtual]

getMaxPosition() implementation of AudioReaderInterface.

See also

AudioReaderInterface::getMaxPosition

Implements AudioReaderInterface.

6.153.3.5 void WavAudioReader::getMediaInfo (MediaInfo & *mediaInfo*) const [virtual]

getMediaInfo() implementation of AudioReaderInterface.

See also

AudioReaderInterface::getMediaInfo

Implements AudioReaderInterface.

6.153.3.6 cardinal WavAudioReader::getNextBlock (void * *buffer*, const cardinal *blockSize*) [virtual]

getNextBlock() implementation of AudioReaderInterface.

See also

AudioReaderInterface::getNextBlock

Implements AudioReaderInterface.

6.153.3.7 card64 WavAudioReader::getPosition ( ) const [virtual]

getPosition() implementation of AudioReaderInterface.

See also

AudioReaderInterface::getPosition

Implements AudioReaderInterface.

6.153.3.8 bool WavAudioReader::openMedia (const char * *name*) [virtual]

openMedia() implementation of AudioReaderInterface.

See also

AudioReaderInterface::openMedia

Implements AudioReaderInterface.

6.153.3.9 bool WavAudioReader::ready ( ) const [virtual]

ready() implementation of AudioReaderInterface.

See also

AudioReaderInterface::ready

Implements AudioReaderInterface.

6.153.3.10 void WavAudioReader::setPosition (const card64 *position*) [virtual]

setPosition() implementation of AudioReaderInterface.

See also

AudioReaderInterface::setPosition

Implements AudioReaderInterface.

6.153.4 Member Data Documentation

6.153.4.1 card64 WavAudioReader::EndPosition [private]

6.153.4.2 MediaError WavAudioReader::Error [private]

6.153.4.3 WAVE_Format WavAudioReader::Format [private]

6.153.4.4 FILE* WavAudioReader::InputFD [private]

6.153.4.5 card64 WavAudioReader::MaxPosition [private]

6.153.4.6 card64 WavAudioReader::Position [private]

6.153.4.7 card64 WavAudioReader::StartPosition [private]

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

- wavaudioreader.h
- wavaudioreader.cc

6.154 WavAudioReader::WAVE_Format Struct Reference

Public Attributes

- card16 FormatTag
- card16 Channels
- card32 SamplesPerSec
- card32 AvgBytesPerSec
- card16 BlockAlign

6.154.1 Member Data Documentation

6.154.1.1 card32 WavAudioReader::WAVE_Format::AvgBytesPerSec

6.154.1.2 card16 WavAudioReader::WAVE_Format::BlockAlign

6.154.1.3 card16 WavAudioReader::WAVE_Format::Channels

6.154.1.4 card16 WavAudioReader::WAVE_Format::FormatTag

6.154.1.5 card32 WavAudioReader::WAVE_Format::SamplesPerSec

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

- wavaudioreader.h

Chapter 7

File Documentation

7.1 abstractlayerdescription.cc File Reference

```
#include "tdsystem.h" #include "abstractlayerdescription.-
h"
```

Defines

- #define USE_FRAMECOUNT_APPROXIMATION

7.1.1 Define Documentation

7.1.1.1 #define USE_FRAMECOUNT_APPROXIMATION

7.2 abstractlayerdescription.h File Reference

```
#include "tdsystem.h" #include "framesizescalabilityinterface.-
h" #include "internetflow.h" #include "abstractlayerdescription.-
icc"
```

Classes

- class AbstractLayerDescription

  Abstract Layer Description.

7.3 abstractqosdescription.cc File Reference

```
#include "tdsystem.h"   #include "abstractqosdescription.-
h"
```

Functions

- std::ostream & operator<< (std::ostream &os, const AbstractQoSDescription &aqd)

7.3.1 Function Documentation

7.3.1.1 std::ostream& operator<< (std::ostream & *os*, const AbstractQoSDescription & *aqd*)

Output operator.

7.4 abstractqosdescription.h File Reference

```
#include "tdsystem.h" #include "frameratescalabilityinterface.-
h" #include "abstractlayerdescription.h" #include "rtppacket.-
h" #include "resourceutilizationpoint.h" #include "abstractqosdescription.-
icc"
```

Classes

- class AbstractQoSDescription

  Abstract QoS Description.

Functions

- std::ostream & operator<< (std::ostream &os, const AbstractQoSDescription &aqd)

7.4.1 Function Documentation

7.4.1.1 std::ostream& operator<< (std::ostream & *os*, const AbstractQoSDescription & *aqd*)

Output operator.

7.5 advancedaudiodecoder.cc File Reference

```
#include "tdsystem.h"    #include "advancedaudiodecoder.h"
#include "advancedaudiopacket.h" #include "audioconverter.-
h"
```

7.6 advancedaudiodecoder.h File Reference

```
#include "tdsystem.h"  #include "audiowriterinterface.h" ×
#include "audiodecoderinterface.h" #include "timedthread.-
h" #include "seqnumvalidator.h" #include "audioquality.h"
#include "advancedaudiopacket.h" #include <set> #include
<map>
```

Classes

- class AdvancedAudioDecoder

  Advanced Audio Decoder.
- struct AdvancedAudioDecoder::FrameFragment
- struct AdvancedAudioDecoder::FrameNode
- struct AdvancedAudioDecoder::FrameNodeItem

7.7 advancedaudioencoder.cc File Reference

```
#include "tdsystem.h"    #include "advancedaudioencoder.h"
#include "advancedaudiopacket.h" #include "audioconverter.-
h" #include "tools.h"
```

7.8 advancedaudioencoder.h File Reference

```
#include "tdsystem.h"  #include "audioencoderinterface.h"
#include "audioreaderinterface.h" #include "audioquality.-
h"
```

Classes

- class AdvancedAudioEncoder

  Advanced Audio Encoder.

Defines

- #define ADVANCEDAUDIOENCDOER_H

7.8.1 Define Documentation

7.8.1.1 #define ADVANCEDAUDIOENCDOER_H

7.9 advancedaudiopacket.cc File Reference

```
#include "tdsystem.h"   #include "advancedaudiopacket.h" ×
#include "tools.h"
```

Classes

- struct Layer
- struct Level

Functions

- static void calculateLevelForQuality (Level &level, const cardinal headerSize, const cardinal maxPacketSize, const AudioQualityInterface &quality)

7.9.1 Function Documentation

7.9.1.1 static void calculateLevelForQuality (Level & *level,* const cardinal *headerSize,* const cardinal *maxPacketSize,* const AudioQualityInterface & *quality*) [static]

7.10 advancedaudiopacket.h File Reference

```
#include "tdsystem.h"   #include "mediainfo.h"   #include
"audioquality.h"
```

Classes

- struct AdvancedAudioPacket

  Advanced Audio Packet.

Enumerations

- enum AdvancedAudioFlags { AAF_ChannelLeft = (1 << 0), AAF_ChannelRight = (1 << 1), AAF_ByteUpper = (1 << 2), AAF_ByteLower = (1 << 3), AAF_-MediaInfo = (1 << 4) }

Functions

- struct AdvancedAudioPacket __attribute__ ((packed))
- AdvancedAudioPacket ()
- void translate ()
- void reset ()
- static AudioQuality calculateQualityForLimits (const AudioQualityInterface &userSetting, const AudioQualityInterface &inputQuality, const card64 totalByteRateLimit, const card64 byteRateLimitL1, const card64 byteRateLimitL2, const card64 byteRateLimitL3, const cardinal networkQualityDecrement, const cardinal headerSize, const cardinal maxPacketSize)
- static cardinal calculateFrameSize (const cardinal inputBytesPerSecond, const cardinal inputFrameSize)
- static cardinal calculateLayers (const AudioQualityInterface &quality)

Variables

- static const card16 AdvancedAudioTypeID = 0x2961
- static const char AdvancedAudioTypeName []
- static const card32 AdvancedAudioFormatID = 0x74660000 | AdvancedAudioTypeID
- static const cardinal AdvancedAudioMediaInfoPacketsPerSecond = 2
- static const cardinal AdvancedAudioFramesPerSecond = 35
- static const cardinal AdvancedAudioFrameSize
- static const cardinal AdvancedAudioMaxTransferDelay = 100 * 16
- static const cardinal AdvancedAudioMaxQualityLayers = 3
- static const cardinal AdvancedAudioQualityLevels = AudioQuality::QualityLevels
- card32 FormatID
- card16 SamplingRate
- card8 Channels
- card8 Bits
- card64 Position
- card64 MaxPosition
- card8 ErrorCode
- card8 Flags
- card16 Fragment
- char Data [0]

7.10.1 Enumeration Type Documentation

7.10.1.1 enum AdvancedAudioFlags

Emumeration of Flags.

Enumerator:

AAF_ChannelLeft

AAF_ChannelRight

AAF_ByteUpper

AAF_ByteLower

AAF_MediaInfo

7.10.2 Function Documentation

7.10.2.1 struct AdvancedAudioPacket __attribute__ ((packed))

7.10.2.2 __attribute__::AdvancedAudioPacket ( )

Constructor.

7.10.2.3 static cardinal __attribute__::calculateFrameSize (const cardinal *inputBytesPerSecond,* const cardinal *inputFrameSize*) [static]

Calculate output frame size from given input bytes per second and input frame size.

Parameters

inputBytes-PerSecond	Input source's bytes per second.
inputFrame-Size	Input source's frame size.

Returns

The calculated frame size.

7.10.2.4 static cardinal __attribute__::calculateLayers (const AudioQualityInterface & *quality*) [static]

Calculate number of layers for given quality.

Parameters

quality	Quality.

Returns

Number of layers.

7.10.2.5 static AudioQuality __attribute__::calculateQualityForLimits (const AudioQualityInterface & *userSetting*, const AudioQualityInterface & *inputQuality*, const card64 *totalByteRateLimit*, const card64 *byteRateLimitL1*, const card64 *byteRateLimitL2*, const card64 *byteRateLimitL3*, const cardinal *networkQualityDecrement*, const cardinal *headerSize*, const cardinal *maxPacketSize*) [static]

Quality calculation for given user quality limited by input quality, byte rate and network quality decrement with given header size (eg. IP + UDP + RTP) and maximum packet size.

Parameters

userSetting	User's quality setting.
inputQuality	Input source's quality.
byteRate-Limit	Byte rate limit.
byteRate-LimitL1	Layer #0 byte rate limit.
byteRate-LimitL2	Layer #1 byte rate limit.
byteRate-LimitL3	Layer #2 byte rate limit.
network-Quality-Decrement	Number of steps for decrement of user's quality.
headerSize	Header size (eg. IP + UDP + RTP). AdvancedAudioPacket size is added automatically.
maxPacket-Size	Maximum packet size.

Returns

The calculated quality.

7.10.2.6 void __attribute__::reset ( )

Reset report.

7.10.2.7 void __attribute__::translate ( )

Translate byte order.

7.10.3 Variable Documentation

7.10.3.1 const card32 AdvancedAudioFormatID = 0x74660000 | AdvancedAudioTypeID [static]

Advanced Audio Encoding package format ID.

7.10.3.2 const cardinal AdvancedAudioFrameSize [static]

Initial value:

```
44100 * 2 * 2 / AdvancedAudioFramesPerSecond
```

Advanced Audio frame size.

7.10.3.3 const cardinal AdvancedAudioFramesPerSecond = 35 [static]

Advanced Audio frames per second.

7.10.3.4 const cardinal AdvancedAudioMaxQualityLayers = 3 [static]

Advanced Audio maximum quality layers.

7.10.3.5 const cardinal AdvancedAudioMaxTransferDelay = 100 * 16 [static]

Advanced Audio maximum transfer delay.

7.10.3.6 const cardinal AdvancedAudioMediaInfoPacketsPerSecond = 2 [static]

Advanced Audio MediaInfo packets per second.

7.10.3.7 const cardinal AdvancedAudioQualityLevels = AudioQuality::QualityLevels [static]

Advanced Audio quality levels.

7.10.3.8 const card16 AdvancedAudioTypeID = 0x2961 [static]

Type ID for Advanced Audio Encoding.

7.10.3.9 const char AdvancedAudioTypeName[] [static]

Name for Advanced Audio Encoding.

7.10.3.10 card8 Bits

Number of audio bits.

7.10.3.11 card8 Channels

Number of audio channels.

7.10.3.12 char Data

Packet data.

7.10.3.13 card8 ErrorCode

Error code.

7.10.3.14 card8 Flags

Advanced Audio Encoding Flags.

7.10.3.15 card32 FormatID

Packet format ID.

7.10.3.16 card16 Fragment

Fragment number.

7.10.3.17 card64 MaxPosition

Maximum position in nanoseconds.

7.10.3.18 card64 Position

Current position in nanoseconds.

7.10.3.19 card16 SamplingRate

Audio sampling rate.

7.11 audioclient.cc File Reference

```
#include "tdsystem.h"  #include "audiodevice.h"  #include
"audiodebug.h" #include "tdsocket.h" #include "simpleaudiodecoder.-
h" #include "advancedaudiodecoder.h" #include "audiodecoderrepository.-
h" #include "rtpreceiver.h" #include "rtcppacket.h" #include
"rtcpsender.h" #include "strings.h" #include "randomizer.-
h" #include "ext_socket.h" #include "audioclientapppacket.-
h" #include "audioclient.h" #include <stdio.h>
```

Defines

- #define DEBUG

7.11.1 Define Documentation

7.11.1.1 #define DEBUG

7.12 audioclient.h File Reference

```
#include "audiowriterinterface.h" #include "audiodecoderinterface.-
h" #include "audiodecoderrepository.h" #include "mediainfo.-
h" #include "rtcpsender.h" #include "rtpreceiver.h" #include
"internetaddress.h" #include "tdsocket.h" #include "strings.-
h"   #include "audioclientapppacket.h"   #include <map> ×
#include "audioclient.icc"
```

Classes

- class AudioClient

  Audio Client.

7.13 audioclientapppacket.cc File Reference

```
#include "tdsystem.h"  #include "audioclientapppacket.h" ×
#include "tools.h"
```

7.14 audioclientapppacket.h File Reference

Classes

- struct AudioClientAppPacket

Audio Client RTCP-APP Packet.

- struct AudioClientSDESPrivPacket

  Audio Client RTCP SDES-PRIV Packet.

Enumerations

- enum AudioClientAppMode { ACAS_UnknownCommand = 0, ACAS_Play = 1, ACAS_Pause = 2 }

Functions

- struct AudioClientAppPacket __attribute__ ((packed))
- AudioClientAppPacket ()
- void translate ()
- void reset ()

Variables

- const card8 AudioServerDefaultTrafficClass = 0x00
- const card8 AudioClientDefaultTrafficClass = 0x00
- static const cardinal RTPAudioDefaultPort = 7500
- static const card32 RTPAudioDataPPID = 0x2909ffff
- static const card32 RTPAudioControlPPID = 0x2909fffe
- static const card32 AudioClientFormatID = 0x75003388
- card32 FormatID
- card16 SequenceNumber
- card16 PosChgSeqNumber
- card16 Status
- card16 SamplingRate
- card8 Channels
- card8 Bits
- card16 Encoding
- card32 BandwidthLimit
- card64 StartPosition
- card64 RestartPosition
- char MediaName [128]
- card8 PrefixLength
- char Prefix [7]

7.14.1 Enumeration Type Documentation

7.14.1.1 enum AudioClientAppMode

Definition of AudioClient commands in APP message.

Enumerator:

ACAS_UnknownCommand

ACAS_Play

ACAS_Pause

7.14.2 Function Documentation

7.14.2.1 struct AudioClientAppPacket __attribute__ ((packed))

7.14.2.2 __attribute__::AudioClientAppPacket ( )

Constructor.

7.14.2.3 void __attribute__::reset ( )

Reset report.

7.14.2.4 void __attribute__::translate ( )

Translate byte order.

7.14.3 Variable Documentation

7.14.3.1 const card8 AudioClientDefaultTrafficClass = 0x00

Default traffic class/TOS for RTCP control connection from client to server.

7.14.3.2 const card32 AudioClientFormatID = 0x75003388 [static]

Packet ID for AudioClient RTCP APP message.

7.14.3.3 const card8 AudioServerDefaultTrafficClass = 0x00

Default traffic class/TOS for RTP data connection from server to client.

7.14.3.4 card32 BandwidthLimit

Suggested bandwidth or 0xffffffff, if unused.

7.14.3.5 card8 Bits

Number of audio bits.

7.14.3.6 card8 Channels

Number of audio channels.

7.14.3.7 card16 Encoding

Encoding.

7.14.3.8 card32 FormatID

Packet ID.

7.14.3.9 char MediaName[128]

Media name, e.g. "AudioFiles/Test1.list".

7.14.3.10 card16 PosChgSeqNumber

Sequence number for position changes.

7.14.3.11 char Prefix[7]

7.14.3.12 card8 PrefixLength

7.14.3.13 card64 RestartPosition

Position to start from if server has been restarted.

7.14.3.14 const card32 RTPAudioControlPPID = 0x2909fffe [static]

RTP Audio control PPID (for SCTP transport).

7.14.3.15 const card32 RTPAudioDataPPID = 0x2909ffff [static]

RTP Audio data PPID (for SCTP transport).

7.14.3.16 const cardinal RTPAudioDefaultPort = 7500 [static]

RTP Audio Server default port.

7.14.3.17 card16 SamplingRate

Audio sampling rate.

7.14.3.18 card16 SequenceNumber

Sequence number.

7.14.3.19 card64 StartPosition

Start position in nanoseconds or 0xffff...ff, if unused.

7.14.3.20 AudioClientAppPacket Status

Client status.

7.15 audioconverter.cc File Reference

```
#include "tdsystem.h" #include "tools.h" #include "audioconverter.-
h"
```

Functions

- cardinal getAlignedLength (const AudioQualityInterface &inputQuality, const AudioQualityInterface &outputQuality, const cardinal inputLength)
- bool getConvParams (const cardinal in, const cardinal out, cardinal &a, cardinal &b, float &c)
- void get12 (const card8 *buffer, card16 &a, card16 &b)
- void set12 (card8 *buffer, const card16 c, const card16 d)
- cardinal AudioConverter (const AudioQualityInterface &from, const AudioQualityInterface &to, const card8 *inputBuffer, card8 *outputBuffer, const cardinal inputLength, const cardinal outputLength)

7.15.1 Function Documentation

7.15.1.1 cardinal AudioConverter (const AudioQualityInterface & *from,* const AudioQualityInterface & *to,* const card8 * *inputBuffer,* card8 * *outputBuffer,* const cardinal *inputLength,* const cardinal *outputLength*)

Audio quality converter. Convert quality from a given value to a given value. Note: The "from" value must be greater than or equal to the "to" value, that is from-sampling rate >= to-sampling rate, from-bits >= to-bits, from-channels >= to-channels.

Parameters

from	Quality to convert from.
to	Quality to convert to.
inputBuffer	Input buffer.
outputBuffer	Output buffer.
inputLength	Length of the audio data in input buffer.
output-Length	Length of the output buffer.

Returns

Length after conversion.

7.15.1.2 void get12 (const card8 * *buffer,* card16 & *a,* card16 & *b*) [inline]

7.15.1.3 cardinal getAlignedLength (const AudioQualityInterface & *inputQuality,* const AudioQualityInterface & *outputQuality,* const cardinal *inputLength*)

Get aligned output length for a conversion from given input quality and input length to output quality. Example: 12 Bit/Stereo has a 6-byte alignment: L1L1R1R2 = 48 bits = 6 Bytes.

Parameters

inputQuality	Input quality.
output-Quality	Output quality.
inputLength	Input length.

Returns

Aligned length.

7.15.1.4 bool getConvParams (const cardinal *in,* const cardinal *out,* cardinal & *a,* cardinal & *b,* float & *c*)

Get parameters for audio conversion. New sampling rate = (a * OldSamplingRate) / b;

Parameters

in	Old sampling rate.
out	New sampling rate.
a	Reference to store a.
b	Reference to store b.
c	Reference to store float in / out.

Returns

true, if a and b have been found; false, if there are no such numbers for b out of the set {1,2,...,20}

7.15.1.5 void set12 (card8 * *buffer,* const card16 *c,* const card16 *d*) [inline]

7.16 audioconverter.h File Reference

```
#include "tdsystem.h" #include "audioquality.h"
```

Functions

- cardinal AudioConverter (const AudioQualityInterface &from, const AudioQualityInterface &to, const card8 *inputBuffer, card8 *outputBuffer, const cardinal inputLength, const cardinal outputLength)
- cardinal getAlignedLength (const AudioQualityInterface &inputQuality, const AudioQualityInterface &outputQuality, const cardinal inputLength)
- bool getConvParams (const cardinal in, const cardinal out, cardinal &a, cardinal &b, float &c)

7.16.1 Function Documentation

7.16.1.1 cardinal AudioConverter (const AudioQualityInterface & *from,* const AudioQualityInterface & *to,* const card8 * *inputBuffer,* card8 * *outputBuffer,* const cardinal *inputLength,* const cardinal *outputLength*)

Audio quality converter. Convert quality from a given value to a given value. Note: The "from" value must be greater than or equal to the "to" value, that is from-sampling rate >= to-sampling rate, from-bits >= to-bits, from-channels >= to-channels.

Parameters

from	Quality to convert from.
to	Quality to convert to.
inputBuffer	Input buffer.
outputBuffer	Output buffer.
inputLength	Length of the audio data in input buffer.
output-Length	Length of the output buffer.

Returns

Length after conversion.

7.16.1.2 cardinal getAlignedLength (const AudioQualityInterface & *inputQuality,* const AudioQualityInterface & *outputQuality,* const cardinal *inputLength*)

Get aligned output length for a conversion from given input quality and input length to output quality. Example: 12 Bit/Stereo has a 6-byte alignment: L1L1R1R2 = 48 bits = 6 Bytes.

Parameters

inputQuality	Input quality.
output-Quality	Output quality.
inputLength	Input length.

Returns

Aligned length.

7.16.1.3 bool getConvParams (const cardinal *in,* const cardinal *out,* cardinal & *a,* cardinal & *b,* float & *c*)

Get parameters for audio conversion. New sampling rate = (a ∗ OldSamplingRate) / b;

Parameters

in	Old sampling rate.
out	New sampling rate.
a	Reference to store a.
b	Reference to store b.
c	Reference to store float in / out.

Returns

true, if a and b have been found; false, if there are no such numbers for b out of the set {1,2,...,20}

7.17 audiodebug.cc File Reference

```
#include "tdsystem.h"  #include "audiodebug.h"  #include
"tools.h" #include "audioquality.h" #include <sys/time.-
h>
```

7.18 audiodebug.h File Reference

```
#include "tdsystem.h" #include "audiowriterinterface.h"
```

Classes

- class AudioDebug

  Audio Debug.

7.19 audiodecoderinterface.h File Reference

```
#include "tdsystem.h" #include "decoderinterface.h" #include
"audioquality.h"
```

Classes

- class AudioDecoderInterface

  Audio Decoder Interface.

7.20 audiodecoderrepository.cc File Reference

```
#include "tdsystem.h"   #include "audiodecoderrepository.-
h"
```

7.21 audiodecoderrepository.h File Reference

```
#include "tdsystem.h"   #include "audiodecoderinterface.h"
#include "decoderrepositoryinterface.h" #include "audioquality.-
h" #include <map> #include "audiodecoderrepository.icc"
```

Classes

- class AudioDecoderRepository

  Audio Decoder Repository.

7.22 audiodevice.cc File Reference

```
#include "tdsystem.h" #include "tools.h" #include "audiodevice.-
h" #include "audioconverter.h" #include <assert.h> #include
```

```
<fcntl.h> #include <sys/ioctl.h> #include <sys/soundcard.-
h>
```

7.23 audiodevice.h File Reference

```
#include "tdsystem.h"    #include "audiowriterinterface.-
h"  #include "thread.h"  #include "ringbuffer.h"  #include
"audiodevice.icc"
```

Classes

- class AudioDevice

  Audio Device.

7.24 audioencoderinterface.h File Reference

```
#include "tdsystem.h" #include "encoderinterface.h" #include
"audioqualityinterface.h"
```

Classes

- class AudioEncoderInterface

  Audio Encoder Interface.

7.25 audioencoderrepository.cc File Reference

```
#include "tdsystem.h"  #include "audioencoderrepository.-
h" #include "audioquality.h"
```

7.26 audioencoderrepository.h File Reference

```
#include "tdsystem.h" #include "synchronizable.h" #include
"encoderrepositoryinterface.h" #include "audioencoderinterface.-
h" #include <map> #include "audioencoderrepository.icc"
```

Classes

- class AudioEncoderRepository

  Audio Encoder Repository.

7.27 audiomixer.cc File Reference

```
#include "tdsystem.h"  #include "audiomixer.h"  #include
<fcntl.h> #include <sys/ioctl.h> #include <sys/types.h>
#include <sys/soundcard.h>
```

7.28 audiomixer.h File Reference

```
#include "tdsystem.h"  #include "audiodevice.h"  #include
<sys/soundcard.h> #include "audiomixer.icc"
```

Classes

- class AudioMixer

  Audio Mixer.

7.29 audionull.cc File Reference

```
#include "tdsystem.h"  #include "audionull.h"  #include
"tools.h" #include <iostream> #include <sys/time.h>
```

7.30 audionull.h File Reference

```
#include "tdsystem.h"  #include "audiowriterinterface.h" ×
#include "audioquality.h"
```

Classes

- class AudioNull

  Audio Null.

7.31 audioquality.cc File Reference

```
#include "tdsystem.h"  #include "audioquality.h"  #include
"randomizer.h"
```

Functions

- std::ostream & operator<< (std::ostream &os, const AudioQualityInterface &quality)

- AudioQuality operator+ (const AudioQualityInterface &q1, const AudioQualityInterface &q2)
- AudioQuality operator- (const AudioQualityInterface &q1, const AudioQualityInterface &q2)
- AudioQuality operator- (const AudioQualityInterface &q1, const cardinal bytesPerSecond)

Variables

- static const card16 _ValidRatesTable []
- static const card8 _ValidBitsTable []
- static const card8 _ValidChannelsTable []

7.31.1 Function Documentation

7.31.1.1 AudioQuality operator+ (const AudioQualityInterface & *q1*, const AudioQualityInterface & *q2*)

Implementation of + operator.

7.31.1.2 AudioQuality operator- (const AudioQualityInterface & *q1*, const AudioQualityInterface & *q2*)

Implementation of - operator.

7.31.1.3 AudioQuality operator- (const AudioQualityInterface & *q1*, const cardinal *bytesPerSecond*)

Implementation of - operator. Limits resulting audio quality by a given byte rate.

7.31.1.4 std::ostream& operator<< (std::ostream & *out*, const AudioQualityInterface & *quality*)

Implementation of << operator.

7.31.2 Variable Documentation

7.31.2.1 const card8 _ValidBitsTable[] [static]

Initial value:

```
 {
   4, 8, 12, 16
}
```

7.31.2.2 const card8 _ValidChannelsTable[] [static]

Initial value:

```
{
  1, 2
}
```

7.31.2.3 const card16 _ValidRatesTable[] [static]

Initial value:

```
{
  4410,   6615,  8820, 11025,
  13230, 15435, 17640, 19845,
  22050, 24255, 26460, 28665,
  30870, 33075, 35280, 37485,
  39690, 41895, 44100
}
```

7.32 audioquality.h File Reference

`#include "tdsystem.h"  #include "randomizer.h"  #include "audioqualityinterface.h" #include "audioquality.icc"`

Classes

- class AudioQuality
  Audio Quality.

Functions

- std::ostream & operator<< (std::ostream &out, const AudioQualityInterface &quality)
- AudioQuality operator+ (const AudioQualityInterface &q1, const AudioQualityInterface &q2)
- AudioQuality operator- (const AudioQualityInterface &q1, const AudioQualityInterface &q2)
- AudioQuality operator- (const AudioQualityInterface &q1, const cardinal bytesPerSecond)

7.32.1 Function Documentation

7.32.1.1 AudioQuality operator+ (const AudioQualityInterface & *q1*, const AudioQualityInterface & *q2*)

Implementation of + operator.

7.32.1.2 AudioQuality operator- (const AudioQualityInterface & *q1*, const AudioQualityInterface & *q2*)

Implementation of - operator.

7.32.1.3 AudioQuality operator- (const AudioQualityInterface & *q1*, const cardinal *bytesPerSecond*)

Implementation of - operator. Limits resulting audio quality by a given byte rate.

7.32.1.4 std::ostream& operator<< (std::ostream & *out*, const AudioQualityInterface & *quality*)

Implementation of << operator.

7.33 audioqualityinterface.h File Reference

```
#include "tdsystem.h"    #include "audioqualityinterface.-
icc"
```

Classes

- class AudioQualityInterface

  Audio Quality Interface.
- class AdjustableAudioQualityInterface

  Adjustable Audio Quality Interface.

7.34 audioreaderinterface.h File Reference

```
#include "tdsystem.h"   #include "audioqualityinterface.h"
#include "mediainfo.h"
```

Classes

- class AudioReaderInterface

  Audio Reader Interface.

7.35 audioserver.cc File Reference

```
#include "tdsystem.h" #include "multiaudioreader.h" #include
"tdsocket.h"    #include "simpleaudioencoder.h"    #include
```

```
"advancedaudioencoder.h" #include "audioencoderrepository.-
h" #include "rtpsender.h" #include "rtcppacket.h" #include
"rtcpreceiver.h"   #include "sourcestateinfo.h"   #include
"tools.h" #include "randomizer.h" #include "audioclientapppacket.-
h" #include "audioserver.h"
```

Defines

- #define VERBOSE

7.35.1 Define Documentation

7.35.1.1 #define VERBOSE

7.36 audioserver.h File Reference

```
#include "tdsystem.h" #include "multiaudioreader.h" #include
"tdsocket.h" #include "audioencoderrepository.h" #include
"rtpsender.h" #include "rtcppacket.h" #include "rtcpreceiver.-
h" #include "rtcpabstractserver.h" #include "qosmanagerinterface.-
h"   #include "audioclientapppacket.h"   #include <map> ×
#include "audioserver.icc"
```

Classes

- class AudioServer

  Audio Server.
- struct AudioServer::User

7.37 audiowriterinterface.h File Reference

```
#include "tdsystem.h" #include "audioqualityinterface.h"
```

Classes

- class AudioWriterInterface

  Audio Writer Interface.

7.38 bandwidthinfo.cc File Reference

```
#include "tdsystem.h" #include "bandwidthinfo.h"
```

Functions

- std::ostream & operator<< (std::ostream &os, const BandwidthInfo &bi)

7.38.1 Function Documentation

7.38.1.1 std::ostream& operator<< (std::ostream & *os,* const BandwidthInfo & *bi*)

Operator "<<".

7.39 bandwidthinfo.h File Reference

`#include "tdsystem.h" #include "bandwidthinfo.icc"`

Classes

- struct BandwidthInfo

  Bandwidth Info.

Functions

- std::ostream & operator<< (std::ostream &os, const BandwidthInfo &bi)

7.39.1 Function Documentation

7.39.1.1 std::ostream& operator<< (std::ostream & *os,* const BandwidthInfo & *bi*)

Operator "<<".

7.40 bandwidthmanager.cc File Reference

`#include "tdsystem.h" #include "bandwidthmanager.h" #include "streamdescription.h"`

Functions

- std::ostream & operator<< (std::ostream &os, const ResourceUtilizationSimplePoint &srup)
- std::ostream & operator<< (std::ostream &os, const ResourceUtilizationMultiPoint &srup)

7.40.1 Function Documentation

7.40.1.1 std::ostream& operator<< (std::ostream & *os,* const ResourceUtilizationSimplePoint & *srup*)

Output operator.

7.40.1.2 std::ostream& operator<< (std::ostream & *os,* const ResourceUtilizationMultiPoint & *srup*)

Output operator.

7.41 bandwidthmanager.h File Reference

```
#include "tdsystem.h"    #include "qosmanagerinterface.h"
#include "abstractqosdescription.h" #include "timedthread.-
h" #include "servicelevelagreement.h" #include "streamdescription.-
h" #include "sessiondescription.h" #include "roundtriptimepinger.-
h" #include "rtcppacket.h" #include <map> #include <algorithm> ×
#include "bandwidthmanager.icc"
```

Classes

- struct ResourceUtilizationSimplePoint

  Resource Utilization Simple Point.
- struct ResourceUtilizationMultiPoint

  Resource Utilization Simple Point.
- class BandwidthManager

  Bandwidth Manager.

Functions

- std::ostream & operator<< (std::ostream &os, const ResourceUtilizationSimple-Point &srup)
- std::ostream & operator<< (std::ostream &os, const ResourceUtilizationMulti-Point &srup)

7.41.1 Function Documentation

7.41.1.1 std::ostream& operator<< (std::ostream & *os,* const ResourceUtilizationSimplePoint & *srup*)

Output operator.

7.41.1.2 std::ostream& operator<< (std::ostream & *os*, const ResourceUtilizationMultiPoint & *srup*)

Output operator.

7.42 breakdetector.cc File Reference

```
#include "tdsystem.h" #include "breakdetector.h" #include
"tools.h" #include <signal.h>
```

Defines

- #define KILL_AFTER_TIMEOUT
- #define KILL_TIMEOUT 2000000

Functions

- void breakDetector (int signum)
- void installBreakDetector ()
- void uninstallBreakDetector ()
- bool breakDetected ()
- void sendBreak (const bool quiet)

Variables

- static bool DetectedBreak = false
- static bool PrintedBreak = false
- static bool Quiet = false
- static pid_t MainThreadPID = getpid()
- static bool PrintedKill = false
- static card64 LastDetection = (card64)-1

7.42.1 Define Documentation

7.42.1.1 #define KILL_AFTER_TIMEOUT

7.42.1.2 #define KILL_TIMEOUT 2000000

7.42.2 Function Documentation

7.42.2.1 bool breakDetected ( )

Check, if break has been detected.

7.42.2.2 void breakDetector (int *signum*)

7.42.2.3 void installBreakDetector ( )

Install break handler.

7.42.2.4 void sendBreak (const bool *quiet* = `false`)

Send break to main thread.

Parameters

quiet	true to print no break message in breakDetected(), false otherwise (default).

7.42.2.5 void uninstallBreakDetector ( )

Uninstall break handler.

7.42.3 Variable Documentation

7.42.3.1 bool DetectedBreak = false `[static]`

7.42.3.2 card64 LastDetection = (card64)-1 `[static]`

7.42.3.3 pid_t MainThreadPID = getpid() `[static]`

7.42.3.4 bool PrintedBreak = false `[static]`

7.42.3.5 bool PrintedKill = false `[static]`

7.42.3.6 bool Quiet = false `[static]`

7.43 breakdetector.h File Reference

```
#include "tdsystem.h"
```

Functions

- void installBreakDetector ()
- void uninstallBreakDetector ()
- bool breakDetected ()
- void sendBreak (const bool quiet=false)

7.43.1 Function Documentation

7.43.1.1 bool breakDetected ( )

Check, if break has been detected.

7.43.1.2 void installBreakDetector ( )

Install break handler.

7.43.1.3 void sendBreak (const bool *quiet* = `false`)

Send break to main thread.

Parameters

quiet	true to print no break message in breakDetected(), false otherwise (default).

7.43.1.4 void uninstallBreakDetector ( )

Uninstall break handler.

7.44 condition.cc File Reference

```
#include "tdsystem.h"   #include "condition.h"   #include
"thread.h" #include <sys/time.h>
```

7.45 condition.h File Reference

```
#include "tdsystem.h" #include "synchronizable.h" #include
"condition.icc"
```

Classes

- class Condition

  Condition.

7.46 decoderinterface.h File Reference

```
#include "tdsystem.h" #include "sourcestateinfo.h" #include
"mediainfo.h"
```

Classes

- struct DecoderPacket

  DecoderPacket.
- class DecoderInterface

  Decoder Interface.

7.47 decoderrepositoryinterface.h File Reference

```
#include "tdsystem.h" #include "synchronizable.h" #include
"decoderinterface.h"
```

Classes

- class DecoderRepositoryInterface

  Decoder Repository.

7.48 encoderinterface.h File Reference

```
#include "tdsystem.h"  #include "abstractqosdescription.-
h"
```

Classes

- struct EncoderPacket

  EncoderPacket.
- class EncoderInterface

  Encoder Interface.

7.49 encoderrepositoryinterface.h File Reference

```
#include "tdsystem.h" #include "synchronizable.h" #include
"encoderinterface.h"
```

Classes

- class EncoderRepositoryInterface

  Encoder Repository Interface.

7.50 ext_socket.h File Reference

```
#include <sys/types.h> #include <sys/socket.h> #include
<sys/uio.h> #include <sys/fcntl.h> #include <sys/time.-
h> #include <inttypes.h> #include <netinet/in.h> #include
<poll.h>
```

Classes

- struct sctp_initmsg
- struct sctp_sndrcvinfo
- struct sctp_assoc_change
- struct sctp_paddr_change
- struct sctp_remote_error
- struct sctp_send_failed
- struct sctp_shutdown_event
- struct sctp_adaptation_event
- struct sctp_pdapi_event
- struct sctp_data_arrive
- union sctp_notification
- struct sctp_rtoinfo
- struct sctp_assocparams
- struct sctp_setprim
- struct sctp_setpeerprim
- struct sctp_setstrm_timeout
- struct sctp_paddrparams
- struct sctp_paddrinfo
- struct sctp_status
- struct sctp_event_subscribe
- struct sctp_assoc_value
- struct sctp_sack_info

Defines

- #define SOCKETAPI_MAJOR_VERSION 0x2
- #define SOCKETAPI_MINOR_VERSION 0x2200
- #define MSG_UNORDERED MSG_DONTROUTE
- #define MSG_UNBUNDLED MSG_CTRUNC
- #define MSG_SHUTDOWN MSG_EOF

- #define MSG_MULTIADDRS MSG_TRUNC
- #define MSG_ABORT 0x200
- #define MSG_PR_SCTP_TTL 0x400
- #define MSG_ADDR_OVER 0x800
- #define MSG_SEND_TO_ALL 0xc00
- #define SCTP_UNORDERED MSG_UNORDERED
- #define SCTP_UNBUNDLED MSG_UNBUNDLED
- #define SCTP_NOTIFICATION MSG_NOTIFICATION
- #define SCTP_ABORT MSG_ABORT
- #define SCTP_EOF MSG_EOF
- #define SCTP_ADDR_OVER MSG_ADDR_OVER
- #define SCTP_SEND_TO_ALL MSG_SEND_TO_ALL
- #define SCTP_MULTIADDRS MSG_MULTIADDRS
- #define SCTP_UNDEFINED 0
- #define SCTP_INIT 1
- #define SCTP_SNDRCV 2
- #define SCTP_ASSOC_CHANGE 1
- #define SCTP_COMM_UP 11
- #define SCTP_COMM_LOST 12
- #define SCTP_RESTART 13
- #define SCTP_SHUTDOWN_COMP 14
- #define SCTP_CANT_STR_ASSOC 15
- #define SCTP_PEER_ADDR_CHANGE 2
- #define SCTP_ADDR_REACHABLE 21
- #define SCTP_ADDR_UNREACHABLE 22
- #define SCTP_ADDR_REMOVED 23
- #define SCTP_ADDR_ADDED 24
- #define SCTP_ADDR_MADE_PRIM 25
- #define SCTP_ADDR_CONFIRMED 26
- #define SCTP_REMOTE_ERROR 3
- #define SCTP_SEND_FAILED 4
- #define SCTP_DATA_UNSENT 41
- #define SCTP_DATA_SENT 42
- #define SCTP_SHUTDOWN_EVENT 5
- #define SCTP_ADAPTATION_INDICATION 6
- #define SCTP_PARTIAL_DELIVERY_EVENT 7
- #define SCTP_PARTIAL_DELIVERY_ABORTED 1
- #define SCTP_DATA_ARRIVE 8
- #define SCTP_ARRIVE_UNORDERED (1 << 0)
- #define SPP_HB_ENABLE (1 << 0)
- #define SPP_HB_DISABLE (1 << 1)
- #define SPP_PMTUD_ENABLE (1 << 2)
- #define SPP_PMTUD_DISABLE (1 << 3)
- #define SPP_SACKDELAY_ENABLE (1 << 4)
- #define SPP_SACKDELAY_DISABLE (1 << 5)
- #define SCTP_INACTIVE 0

- #define SCTP_ACTIVE 1
- #define SCTP_INITMSG 1000
- #define SCTP_AUTOCLOSE 1001
- #define SCTP_RTOINFO 1010
- #define SCTP_ASSOCINFO 1011
- #define SCTP_PRIMARY_ADDR 1012
- #define SCTP_SET_PEER_PRIMARY_ADDR 1013
- #define SCTP_SET_STREAM_TIMEOUTS 1014
- #define SCTP_PEER_ADDR_PARAMS 1015
- #define SCTP_STATUS 1016
- #define SCTP_GET_PEER_ADDR_INFO 1017
- #define SCTP_NODELAY 1018
- #define SCTP_SET_DEFAULT_SEND_PARAM 1019
- #define SCTP_EVENTS 1020
- #define SCTP_DELAYED_SACK 1021
- #define SCTP_FRAGMENT_INTERLEAVE 1022
- #define SCTP_PARTIAL_DELIVERY_POINT 1023
- #define SCTP_MAXSEG 1024
- #define SCTP_I_WANT_MAPPED_V4_ADDR 1025
- #define SCTP_BINDX_ADD_ADDR 1
- #define SCTP_BINDX_REM_ADDR 2
- #define sctp_connectx ext_connectx

Typedefs

- typedef unsigned int sctp_assoc_t
- typedef unsigned short sctp_stream_t

Functions

- unsigned int socketAPIGetVersion ()
- int ext_socket (int domain, int type, int protocol)
- int ext_open (const char *pathname, int flags, mode_t mode)
- int ext_creat (const char *pathname, mode_t mode)
- int ext_bind (int sockfd, struct sockaddr *my_addr, socklen_t addrlen)
- int ext_connect (int sockfd, const struct sockaddr *serv_addr, socklen_t addrlen)
- int ext_listen (int s, int backlog)
- int ext_accept (int s, struct sockaddr *addr, socklen_t *addrlen)
- int ext_shutdown (int s, int how)
- int ext_close (int fd)
- int ext_getsockname (int sockfd, struct sockaddr *name, socklen_t *namelen)
- int ext_getpeername (int sockfd, struct sockaddr *name, socklen_t *namelen)
- int ext_fcntl (int fd, int cmd,...)
- int ext_ioctl (int d, int request, const void *argp)
- int ext_getsockopt (int sockfd, int level, int optname, void *optval, socklen_t *optlen)

- int ext_setsockopt (int sockfd, int level, int optname, const void *optval, socklen_t optlen)
- ssize_t ext_recv (int s, void *buf, size_t len, int flags)
- ssize_t ext_recvfrom (int s, void *buf, size_t len, int flags, struct sockaddr *from, socklen_t *fromlen)
- ssize_t ext_recvmsg (int s, struct msghdr *msg, int flags)
- ssize_t ext_send (int s, const void *msg, size_t len, int flags)
- ssize_t ext_sendto (int s, const void *msg, size_t len, int flags, const struct sockaddr *to, socklen_t tolen)
- ssize_t ext_sendmsg (int s, const struct msghdr *msg, int flags)
- ssize_t ext_read (int fd, void *buf, size_t count)
- ssize_t ext_write (int fd, const void *buf, size_t count)
- int ext_select (int n, fd_set *readfds, fd_set *writefds, fd_set *exceptfds, struct timeval *timeout)
- int ext_poll (struct pollfd *fdlist, long unsigned int count, int time)
- int ext_recvmsg2 (int sockfd, struct msghdr *msg, int flags, const int receiveNotifications)
- int sctp_bindx (int sockfd, struct sockaddr *addrs, int addrcnt, int flags)
- int ext_connectx (int sockfd, const struct sockaddr *addrs, int addrcnt, sctp_assoc_t *id)
- int sctp_peeloff (int sockfd, sctp_assoc_t id)
- int sctp_getpaddrs (int sockfd, sctp_assoc_t id, struct sockaddr **addrs)
- void sctp_freepaddrs (struct sockaddr *addrs)
- int sctp_getladdrs (int sockfd, sctp_assoc_t id, struct sockaddr **addrs)
- void sctp_freeladdrs (struct sockaddr *addrs)
- int sctp_opt_info (int sd, sctp_assoc_t assocID, int opt, void *arg, socklen_t *size)
- ssize_t sctp_send (int s, const void *data, size_t len, const struct sctp_sndrcvinfo *sinfo, int flags)
- ssize_t sctp_sendx (int sd, const void *data, size_t len, const struct sockaddr *addrs, int addrcnt, const struct sctp_sndrcvinfo *sinfo, int flags)
- ssize_t sctp_sendmsg (int s, const void *data, size_t len, const struct sockaddr *to, socklen_t tolen, uint32_t ppid, uint32_t flags, uint16_t stream_no, uint32_t timetolive, uint32_t context)
- ssize_t sctp_recvmsg (int s, void *msg, size_t len, struct sockaddr *from, socklen_t *fromlen, struct sctp_sndrcvinfo *sinfo, int *msg_flags)
- int ext_pipe (int fds[2])
- int sctp_isavailable ()
- int sctp_enableOOTBHandling (const unsigned int enable)
- int sctp_enableCRC32 (const unsigned int enable)

7.50.1 Define Documentation

7.50.1.1 #define MSG_ABORT 0x200

7.50.1.2 #define MSG_ADDR_OVER 0x800

7.50.1.3 #define MSG_MULTIADDRS MSG_TRUNC

7.50.1.4 #define MSG_PR_SCTP_TTL 0x400

7.50.1.5 #define MSG_SEND_TO_ALL 0xc00

7.50.1.6 #define MSG_SHUTDOWN MSG_EOF

7.50.1.7 #define MSG_UNBUNDLED MSG_CTRUNC

7.50.1.8 #define MSG_UNORDERED MSG_DONTROUTE

7.50.1.9 #define SCTP_ABORT MSG_ABORT

7.50.1.10 #define SCTP_ACTIVE 1

7.50.1.11 #define SCTP_ADAPTATION_INDICATION 6

7.50.1.12 #define SCTP_ADDR_ADDED 24

7.50.1.13 #define SCTP_ADDR_CONFIRMED 26

7.50.1.14 #define SCTP_ADDR_MADE_PRIM 25

7.50.1.15 #define SCTP_ADDR_OVER MSG_ADDR_OVER

7.50.1.16 #define SCTP_ADDR_REACHABLE 21

7.50.1.17 #define SCTP_ADDR_REMOVED 23

7.50.1.18 #define SCTP_ADDR_UNREACHABLE 22

7.50.1.19 #define SCTP_ARRIVE_UNORDERED (1 << 0)

7.50.1.20 #define SCTP_ASSOC_CHANGE 1

7.50.1.21 #define SCTP_ASSOCINFO 1011

7.50.1.22 #define SCTP_AUTOCLOSE 1001

7.50.1.23 #define SCTP_BINDX_ADD_ADDR 1

7.50.1.24 #define SCTP_BINDX_REM_ADDR 2

7.50.1.25 #define SCTP_CANT_STR_ASSOC 15

7.50.1.26 #define SCTP_COMM_LOST 12

7.50.1.27 #define SCTP_COMM_UP 11

7.50.1.28 #define sctp_connectx ext_connectx

7.50.1.29 #define SCTP_DATA_ARRIVE 8

7.50.1.30 #define SCTP_DATA_SENT 42

7.50.1.31 #define SCTP_DATA_UNSENT 41

7.50.1.32 #define SCTP_DELAYED_SACK 1021

7.50.1.33 #define SCTP_EOF MSG_EOF

7.50.1.34 #define SCTP_EVENTS 1020

7.50.1.35 #define SCTP_FRAGMENT_INTERLEAVE 1022

7.50.1.36 #define SCTP_GET_PEER_ADDR_INFO 1017

7.50.1.37 #define SCTP_I_WANT_MAPPED_V4_ADDR 1025

7.50.1.38 #define SCTP_INACTIVE 0

7.50.1.39 #define SCTP_INIT 1

7.50.1.40 #define SCTP_INITMSG 1000

7.50.1.41 #define SCTP_MAXSEG 1024

7.50.1.42 #define SCTP_MULTIADDRS MSG_MULTIADDRS

7.50.1.43 #define SCTP_NODELAY 1018

7.50.1.44 #define SCTP_NOTIFICATION MSG_NOTIFICATION

7.50.1.45 #define SCTP_PARTIAL_DELIVERY_ABORTED 1

7.50.1.46 #define SCTP_PARTIAL_DELIVERY_EVENT 7

7.50.1.47 #define SCTP_PARTIAL_DELIVERY_POINT 1023

7.50.1.48 #define SCTP_PEER_ADDR_CHANGE 2

7.50.1.49 #define SCTP_PEER_ADDR_PARAMS 1015

7.50.1.50 #define SCTP_PRIMARY_ADDR 1012

7.50.1.51 #define SCTP_REMOTE_ERROR 3

7.50.1.52 #define SCTP_RESTART 13

7.50.1.53 #define SCTP_RTOINFO 1010

7.50.1.54 #define SCTP_SEND_FAILED 4

7.50.1.55 #define SCTP_SEND_TO_ALL MSG_SEND_TO_ALL

7.50.1.56 #define SCTP_SET_DEFAULT_SEND_PARAM 1019

7.50.1.57 #define SCTP_SET_PEER_PRIMARY_ADDR 1013

7.50.1.58 #define SCTP_SET_STREAM_TIMEOUTS 1014

7.50.1.59 #define SCTP_SHUTDOWN_COMP 14

7.50.1.60 #define SCTP_SHUTDOWN_EVENT 5

7.50.1.61 #define SCTP_SNDRCV 2

7.50.1.62 #define SCTP_STATUS 1016

7.50.1.63 #define SCTP_UNBUNDLED MSG_UNBUNDLED

7.50.1.64 #define SCTP_UNDEFINED 0

7.50.1.65 #define SCTP_UNORDERED MSG_UNORDERED

7.50.1.66 #define SOCKETAPI_MAJOR_VERSION 0x2

7.50.1.67 #define SOCKETAPI_MINOR_VERSION 0x2200

7.50.1.68 #define SPP_HB_DISABLE (1 << 1)

7.50.1.69 #define SPP_HB_ENABLE (1 << 0)

7.50.1.70 #define SPP_PMTUD_DISABLE (1 << 3)

7.50.1.71 #define SPP_PMTUD_ENABLE (1 << 2)

7.50.1.72 #define SPP_SACKDELAY_DISABLE (1 << 5)

7.50.1.73 #define SPP_SACKDELAY_ENABLE (1 << 4)

7.50.2 Typedef Documentation

7.50.2.1 typedef unsigned int **sctp_assoc_t**

7.50.2.2 typedef unsigned short **sctp_stream_t**

7.50.3 Function Documentation

7.50.3.1 int **ext_accept** (int *s,* struct sockaddr * *addr,* socklen_t * *addrlen*)

7.50.3.2 int **ext_bind** (int *sockfd,* struct sockaddr * *my_addr,* socklen_t *addrlen*)

7.50.3.3 int **ext_close** (int *fd*)

7.50.3.4 int **ext_connect** (int *sockfd,* const struct sockaddr * *serv_addr,* socklen_t *addrlen*)

7.50.3.5 int **ext_connectx** (int *sockfd,* const struct sockaddr * *addrs,* int *addrcnt,* **sctp_assoc_t** * *id*)

7.50.3.6 int **ext_creat** (const char * *pathname,* mode_t *mode*)

7.50.3.7 int **ext_fcntl** (int *fd,* int *cmd, ...*)

7.50.3.8 int **ext_getpeername** (int *sockfd,* struct sockaddr * *name,* socklen_t * *namelen*)

7.50.3.9 int **ext_getsockname** (int *sockfd,* struct sockaddr * *name,* socklen_t * *namelen*)

7.50.3.10 int **ext_getsockopt** (int *sockfd,* int *level,* int *optname,* void * *optval,* socklen_t * *optlen*)

7.50.3.11 int **ext_ioctl** (int *d,* int *request,* const void * *argp*)

7.50.3.12 int **ext_listen** (int *s,* int *backlog*)

7.50.3.13 int **ext_open** (const char * *pathname,* int *flags,* mode_t *mode*)

7.50.3.14 int **ext_pipe** (int *fds[2]*)

7.50.3.15 int **ext_poll** (struct pollfd * *fdlist,* long unsigned int *count,* int *time*)

7.50.3.16 ssize_t **ext_read** (int *fd,* void * *buf,* size_t *count*)

7.50.3.17 ssize_t **ext_recv** (int *s,* void * *buf,* size_t *len,* int *flags*)

7.50.3.18 ssize_t **ext_recvfrom** (int *s,* void * *buf,* size_t *len,* int *flags,* struct sockaddr * *from,* socklen_t * *fromlen*)

7.50.3.19 ssize_t **ext_recvmsg** (int *s,* struct msghdr * *msg,* int *flags*)

7.50.3.20 int **ext_recvmsg2** (int *sockfd,* struct msghdr * *msg,* int *flags,* const int *receiveNotifications*)

7.50.3.21 int **ext_select** (int *n,* fd_set * *readfds,* fd_set * *writefds,* fd_set * *exceptfds,* struct timeval * *timeout*)

7.50.3.22 ssize_t **ext_send** (int *s,* const void * *msg,* size_t *len,* int *flags*)

7.50.3.23 ssize_t **ext_sendmsg** (int *s,* const struct msghdr * *msg,* int *flags*)

7.50.3.24 ssize_t **ext_sendto** (int *s,* const void * *msg,* size_t *len,* int *flags,* const struct sockaddr * *to,* socklen_t *tolen*)

7.50.3.25 int **ext_setsockopt** (int *sockfd,* int *level,* int *optname,* const void * *optval,* socklen_t *optlen*)

7.50.3.26 int **ext_shutdown** (int *s,* int *how*)

7.50.3.27 int **ext_socket** (int *domain,* int *type,* int *protocol*)

7.50.3.28 ssize_t **ext_write** (int *fd,* const void * *buf,* size_t *count*)

7.50.3.29 int **sctp_bindx** (int *sockfd,* struct sockaddr * *addrs,* int *addrcnt,* int *flags*)

7.50.3.30 int **sctp_enableCRC32** (const unsigned int *enable*)

Enable or disable CRC32 checksum.

Parameters

enable	0 to disable (use Adler32), <>0 to enable CRC32.

Returns

0 for success, error code in case of error.

7.50.3.31 int **sctp_enableOOTBHandling** (const unsigned int *enable*)

Enable or disable OOTB handling.

Parameters

enable	0 to disable, <>0 to enable OOTB handling.

Returns

0 for success, error code in case of error.

7.50.3.32 void **sctp_freeladdrs** (struct sockaddr * *addrs*)

7.50.3.33 void **sctp_freepaddrs** (struct sockaddr * *addrs*)

7.50.3.34 int **sctp_getladdrs** (int *sockfd,* **sctp_assoc_t** *id,* struct sockaddr ** *addrs*)

7.50.3.35 int **sctp_getpaddrs** (int *sockfd,* **sctp_assoc_t** *id,* struct sockaddr ** *addrs*)

7.50.3.36 int **sctp_isavailable** ( )

Check, if SCTP support is available.

Returns

true, if SCTP is available; false otherwise.

7.50.3.37 int **sctp_opt_info** (int *sd,* **sctp_assoc_t** *assocID,* int *opt,* void * *arg,* socklen_t * *size*)

7.50.3.38 int **sctp_peeloff** (int *sockfd,* **sctp_assoc_t** *id*)

7.50.3.39 ssize_t **sctp_recvmsg** (int *s,* void * *msg,* size_t *len,* struct sockaddr * *from,* socklen_t * *fromlen,* struct **sctp_sndrcvinfo** * *sinfo,* int * *msg_flags*)

7.50.3.40 ssize_t **sctp_send** (int *s,* const void * *data,* size_t *len,* const struct **sctp_sndrcvinfo** * *sinfo,* int *flags*)

7.50.3.41 ssize_t **sctp_sendmsg** (int *s,* const void * *data,* size_t *len,* const struct sockaddr * *to,* socklen_t *tolen,* uint32_t *ppid,* uint32_t *flags,* uint16_t *stream_no,* uint32_t *timetolive,* uint32_t *context*)

7.50.3.42 ssize_t **sctp_sendx** (int *sd,* const void * *data,* size_t *len,* const struct sockaddr * *addrs,* int *addrcnt,* const struct **sctp_sndrcvinfo** * *sinfo,* int *flags*)

7.50.3.43 unsigned int **socketAPIGetVersion** ( )

7.51 fft.cc File Reference

```
#include "tdsystem.h" #include "fft.h" #include "tools.h" ×
#include <iostream> #include <sys/time.h>
```

7.52 fft.h File Reference

```
#include "tdsystem.h" #include "audiowriterinterface.h"
```

Classes

- class FastFourierTransformation

  Fast Fourier Transformation.

7.53 frameratescalabilityinterface.h File Reference

```
#include "tdsystem.h"
```

Classes

- class FrameRateScalabilityInterface

  Frame Rate Scalability Interface.

7.54 framesizescalabilityinterface.h File Reference

```
#include "tdsystem.h"
```

Classes

- class FrameSizeScalabilityInterface

  Frame Rate Scalability Interface.

7.55 internetaddress.cc File Reference

```
#include "tdsystem.h" #include "internetaddress.h" #include
"ext_socket.h" #include <netdb.h> #include <netinet/in.-
h> #include <sys/utsname.h> #include <net/if.h> #include
<arpa/nameser.h> #include <ctype.h>
```

7.56 internetaddress.h File Reference

```
#include "tdsystem.h" #include "strings.h" #include "socketaddress.-
h" #include "portableaddress.h" #include <netinet/in.h> ×
#include <resolv.h> #include "internetaddress.icc"
```

Classes

- class InternetAddress

  Socket Address.

7.57 internetflow.cc File Reference

```
#include "tdsystem.h" #include "internetaddress.h" #include
"internetflow.h"
```

7.58 internetflow.h File Reference

```
#include "tdsystem.h" #include "internetaddress.h" #include
"internetflow.icc"
```

Classes

- class InternetFlow

  Internet Flow.

7.59 managedstreaminterface.h File Reference

```
#include "tdsystem.h"   #include "abstractqosdescription.-
h"
```

Classes

- class ManagedStreamInterface

  Managed Stream Interface.

7.60 mediainfo.cc File Reference

```
#include "tdsystem.h"   #include "mediainfo.h"   #include
"tools.h"
```

Functions

- std::ostream & operator<< (std::ostream &os, const MediaInfo &mi)

7.60.1 Function Documentation

7.60.1.1 std::ostream& operator<< (std::ostream & *os,* const MediaInfo & *mi*)

Output operator.

7.61 mediainfo.h File Reference

```
#include "tools.h"
```

Classes

- struct MediaInfo

  Media Info.

Enumerations

- enum MediaError { ME_NoError = 0, ME_NoMedia = 1, ME_EOF = 2, ME_UnrecoverableError = 20, ME_BadMedia = ME_UnrecoverableError + 0, ME_ReadError = ME_UnrecoverableError + 1, ME_OutOfMemory = ME_UnrecoverableError + 2 }

Functions

- struct MediaInfo __attribute ((packed))
- MediaInfo ()
- void reset ()
- void translate ()
- std::ostream & operator<< (std::ostream &os, const MediaInfo &mi)

Variables

- const card64 PositionStepsPerSecond = (card64)1000000000
- card64 StartTimeStamp
- card64 EndTimeStamp
- static const cardinal MaxTitleLength = 47
- static const cardinal MaxArtistLength = 47
- static const cardinal MaxCommentLength = 47
- char Title [MaxTitleLength+1]
- char Artist [MaxArtistLength+1]
- char Comment [MaxCommentLength+1]

7.61.1 Enumeration Type Documentation

7.61.1.1 enum MediaError

Definition of encoder errors.

Enumerator:

ME_NoError

ME_NoMedia

ME_EOF

ME_UnrecoverableError

ME_BadMedia

ME_ReadError

ME_OutOfMemory

7.61.2 Function Documentation

7.61.2.1 struct MediaInfo __attribute ((packed))

7.61.2.2 __attribute::MediaInfo ( )

Constructor.

7.61.2.3 std::ostream& operator<< (std::ostream & *os,* const MediaInfo & *mi*)

Output operator.

7.61.2.4 void __attribute::reset ( )

Reset.

7.61.2.5 void __attribute::translate ( )

Translate byte order.

7.61.3 Variable Documentation

7.61.3.1 char Artist[MaxArtistLength+1]

Artist string.

7.61.3.2 char Comment[MaxCommentLength+1]

Comment string.

7.61.3.3 card64 EndTimeStamp

End time stamp of the media.

7.61.3.4 const cardinal MaxArtistLength = 47 [static]

Constant for the maximum author length.

7.61.3.5 const cardinal MaxCommentLength = 47 [static]

Constant for the maximum comment length.

7.61.3.6 const cardinal MaxTitleLength = 47 [static]

Constant for the maximum title length.

7.61.3.7 const card64 PositionStepsPerSecond = (card64)1000000000

Constant for position steps per second: 1 step = 1 nanosecond;

7.61.3.8 card64 StartTimeStamp

Start time stamp of the media.

7.61.3.9 char Title[MaxTitleLength+1]

Title string.

7.62 mp3audioreader.cc File Reference

```
#include "tdsystem.h" #include "mp3audioreader.h" #include
<stdarg.h>
```

Functions

- void debug (const char *fmt,...)

7.62.1 Function Documentation

7.62.1.1 void debug (const char * *fmt*, ...)

7.63 mp3audioreader.h File Reference

```
#include "tdsystem.h"  #include "audioreaderinterface.h" ×
#include "audioquality.h" #include "mpegsound.h"
```

Classes

- class MP3AudioReader

  MP3 Audio Reader.

Defines

- #define PTHREADEDMPEG 1
- #define HAVE_PTHREAD_H

Variables

- MP3AudioReader __attribute

7.63.1 Define Documentation

7.63.1.1 #define HAVE_PTHREAD_H

7.63.1.2 #define PTHREADEDMPEG 1

7.63.2 Variable Documentation

7.63.2.1 MP3AudioReader __attribute

7.64 multiaudioreader.cc File Reference

```
#include "tdsystem.h" #include "multiaudioreader.h" #include
"wavaudioreader.h" #include "mp3audioreader.h"
```

7.65 multiaudioreader.h File Reference

```
#include "tdsystem.h"     #include "audioreaderinterface.-
h" #include "audioquality.h" #include "string.h" #include
<map>
```

Classes

- class MultiAudioReader

  Multi Audio Reader.
- struct MultiAudioReader::ReaderEntry

7.66 multiaudiowriter.cc File Reference

```
#include "tdsystem.h" #include "multiaudiowriter.h" #include
"tools.h" #include <iostream> #include <sys/time.h>
```

7.67 multiaudiowriter.h File Reference

```
#include "tdsystem.h"  #include "audiowriterinterface.h" ×
#include "synchronizable.h" #include <set>
```

Classes

- class MultiAudioWriter

  Multi Audio Writer.

7.68 multitimerthread.h File Reference

```
#include "tdsystem.h" #include "tools.h" #include "thread.-
h" #include "multitimerthread.icc"
```

Classes

- class MultiTimerThread< Timers >

  Multi Timer Thread.
- struct MultiTimerThread< Timers >::TimerParameters

Typedefs

- typedef MultiTimerThread< 1 > SingleTimerThread

7.68.1 Typedef Documentation

7.68.1.1 typedef MultiTimerThread<1> SingleTimerThread

7.69 packetaddress.cc File Reference

```
#include "tdsystem.h" #include "strings.h" #include "packetaddress.-
h"
```

7.70 packetaddress.h File Reference

```
#include "tdsystem.h" #include "strings.h" #include "socketaddress.-
h" #include <linux/if.h> #include "packetaddress.icc"
```

Classes

- class PacketAddress

  Packet Address.

7.71 pingerhost.h File Reference

```
#include "tdsystem.h" #include "tdsocket.h" #include "strings.-
h"  #include "internetaddress.h"  #include "timedthread.h"
#include "pingerhost.icc"
```

Classes

- struct PingerHost

  PingerHost.

Functions

- int operator== (const PingerHost &ph1, const PingerHost &ph2)
- int operator< (const PingerHost &ph1, const PingerHost &ph2)
- int operator> (const PingerHost &ph1, const PingerHost &ph2)

7.71.1 Function Documentation

7.71.1.1 int operator< (const PingerHost & *ph1,* const PingerHost & *ph2*) [inline]

Operator "<".

7.71.1.2 int operator== (const PingerHost & *ph1,* const PingerHost & *ph2*) [inline]

Operator "==".

Operator "!=".

7.71.1.3 int operator> (const PingerHost & *ph1,* const PingerHost & *ph2*) [inline]

Operator ">".

7.72 portableaddress.h File Reference

```
#include "tdsystem.h" #include "portableaddress.icc"
```

Classes

- class PortableAddress

  Portable Internet Address.

7.73 qaudiomixer.cc File Reference

```
#include "tdsystem.h"  #include "qaudiomixer.h"  #include
"qaudiomixer_moc.cc"  #include "audiodevice.h"  #include
"audiodebug.h" #include "audioconverter.h" #include "timedthread.-
h" #include <qapplication.h> #include <qlayout.h> #include
<qpushbutton.h>  #include <qlabel.h>  #include <qslider.-
h>  #include <qgroupbox.h>  #include <qmainwindow.h> ×
#include <qtimer.h>
```

7.74 qaudiomixer.h File Reference

```
#include "tdsystem.h"  #include "audiomixer.h"  #include
<qapplication.h> #include <qlayout.h> #include <qpushbutton.-
h>  #include <qlabel.h>  #include <qslider.h>  #include
<qgroupbox.h> #include <qmainwindow.h>
```

Classes

- class QAudioMixer

  QAudioMixer.

7.75 qaudiomixer_moc.cc File Reference

```
#include "qaudiomixer.h"
```

Variables

- static QT_BEGIN_MOC_NAMESPACE const uint qt_meta_data_QAudioMixer []
- static const char qt_meta_stringdata_QAudioMixer []

7.75.1 Variable Documentation

7.75.1.1 QT_BEGIN_MOC_NAMESPACE const uint qt_meta_data_QAudioMixer[] [static]

Initial value:

```
 {

       5,
       0,
       0,    0,
       6,   14,
       0,    0,
       0,    0,
       0,    0,
       0,
       1,


      13,   12,   12,   12, 0x05,


      37,   31,   12,   12, 0x0a,
      50,   31,   12,   12, 0x0a,
      62,   12,   12,   12, 0x0a,
      87,   12,   12,   12, 0x0a,
     103,   12,   12,   12, 0x0a,

       0
}
```

7.75.1.2 const char qt_meta_stringdata_QAudioMixer[] [static]

Initial value:

```
 {
    "QAudioMixer\0\0closeAudioMixer()\0value\0"
    "balance(int)\0volume(int)\0"
    "updateVolumeFromDevice()\0centerBalance()\0"
    "mute()\0"
}
```

7.76 qinfotabwidget.cc File Reference

```
#include "tdsystem.h" #include "qinfotabwidget.h" #include
"qinfotabwidget_moc.cc" #include <qapplication.h> #include
<qlayout.h> #include <qpushbutton.h> #include <qtabwidget.-
h> #include <qlabel.h> #include <qlist.h>
```

7.77 qinfotabwidget.h File Reference

```
#include "tdsystem.h" #include <qapplication.h> #include
<qlayout.h> #include <qpushbutton.h> #include <qtabwidget.-
h> #include <qwhatsthis.h> #include <qlabel.h> #include
<qlist.h> #include <qhash.h>
```

Classes

- struct InfoEntry

  InfoEntry.
- struct InfoTable

  InfoTable.
- class QInfoWidget

  QInfoWidget.
- class QInfoTabWidget

  QInfoTabWidget.

7.78 qinfotabwidget_moc.cc File Reference

```
#include "qinfotabwidget.h"
```

Variables

- static QT_BEGIN_MOC_NAMESPACE const uint qt_meta_data_QInfoWidget []
- static const char qt_meta_stringdata_QInfoWidget []
- static const uint qt_meta_data_QInfoTabWidget []
- static const char qt_meta_stringdata_QInfoTabWidget []

7.78.1 Variable Documentation

7.78.1.1 const uint qt_meta_data_QInfoTabWidget[] [static]

Initial value:

```
{

       5,
       0,
       0,    0,
       0,    0,
       0,    0,
       0,    0,
       0,    0,
       0,
```

```
        0,

        0
}
```

7.78.1.2 QT_BEGIN_MOC_NAMESPACE const uint qt_meta_data_QInfoWidget[] [static]

Initial value:

```
 {

        5,
        0,
        0,    0,
        0,    0,
        0,    0,
        0,    0,
        0,    0,
        0,
        0,

        0
}
```

7.78.1.3 const char qt_meta_stringdata_QInfoTabWidget[] [static]

Initial value:

```
 {
    "QInfoTabWidget\0"
}
```

7.78.1.4 const char qt_meta_stringdata_QInfoWidget[] [static]

Initial value:

```
 {
    "QInfoWidget\0"
}
```

7.79 qosmanagerinterface.h File Reference

```
#include "tdsystem.h"   #include "abstractqosdescription.-
h" #include "managedstreaminterface.h" #include "rtcppacket.-
h"
```

Classes

- class QoSManagerInterface

7.80 qspectrumanalyzer.cc File Reference

```
#include "tdsystem.h"     #include "qspectrumanalyzer.h" ×
#include "qspectrumanalyzer_moc.cc" #include "audiodevice.-
h"    #include "audiodebug.h"    #include "audioconverter.-
h"   #include "timedthread.h"   #include <qapplication.h> ×
#include <qlayout.h>   #include <qpushbutton.h>   #include
<qbuttongroup.h> #include <qradiobutton.h> #include <qtimer.-
h> #include <qpainter.h> #include <qgroupbox.h> #include
<qcheckbox.h> #include <qmainwindow.h>
```

7.81 qspectrumanalyzer.h File Reference

```
#include "tdsystem.h" #include "spectrumanalyzer.h" #include
<qapplication.h> #include <qlayout.h> #include <qpushbutton.-
h>   #include <qbuttongroup.h>   #include <qradiobutton.-
h>  #include <qcheckbox.h>  #include <qtimer.h>  #include
<qpainter.h> #include <qgroupbox.h> #include <qmainwindow.-
h>
```

Classes

- class QSpectrumDisplay

  QSpectrumAnalyzer.
- class QSpectrumAnalyzer

  QSpectrumAnalyzer.

Variables

- const card16 QSpectrumAnalyzerTimings []

7.81.1 Variable Documentation

7.81.1.1 const card16 QSpectrumAnalyzerTimings[]

Initial value:

```
 {
   50,100,150,250,350,500,750
}
```

Constants for the timing radio buttons.

7.82 qspectrumanalyzer_moc.cc File Reference

```
#include "qspectrumanalyzer.h"
```

Variables

- static QT_BEGIN_MOC_NAMESPACE const uint qt_meta_data_QSpectrum-Display []
- static const char qt_meta_stringdata_QSpectrumDisplay []
- static const uint qt_meta_data_QSpectrumAnalyzer []
- static const char qt_meta_stringdata_QSpectrumAnalyzer []

7.82.1 Variable Documentation

7.82.1.1 const uint qt_meta_data_QSpectrumAnalyzer[] [static]

Initial value:

```
{

       5,
       0,
       0,    0,
       7,   14,
       0,    0,
       0,    0,
       0,    0,
       0,
       1,


      19,   18,   18,   18, 0x05,


      43,   18,   18,   18, 0x0a,
      59,   56,   18,   18, 0x0a,
      71,   18,   18,   18, 0x0a,
      79,   18,   18,   18, 0x0a,
      99,   93,   18,   18, 0x0a,
     123,  116,   18,   18, 0x0a,

       0
}
```

7.82.1.2 QT_BEGIN_MOC_NAMESPACE const uint qt_meta_data_QSpectrumDisplay[] [static]

Initial value:

```
 {

       5,
       0,
       0,    0,
       2,   14,
       0,    0,
       0,    0,
       0,    0,
       0,
       0,


      18,   17,   17,   17, 0x0a,
      59,   43,   17,   17, 0x0a,

       0
}
```

7.82.1.3 const char qt_meta_stringdata_QSpectrumAnalyzer[] [static]

Initial value:

```
 {
    "QSpectrumAnalyzer\0\0closeSpectrumAnalyzer()\0"
    "timerEvent()\0on\0pause(bool)\0reset()\0"
    "closeWindow()\0index\0newInterval(int)\0"
    "status\0drawAverageLineToggled(int)\0"
}
```

7.82.1.4 const char qt_meta_stringdata_QSpectrumDisplay[] [static]

Initial value:

```
 {
    "QSpectrumDisplay\0\0paintEvent(QPaintEvent*)\0"
    "drawAverageLine\0setDrawAverageLine(bool)\0"
}
```

7.83 randomizer.cc File Reference

```
#include "tdsystem.h"   #include "randomizer.h"   #include
"tools.h"
```

7.84 randomizer.h File Reference

```
#include "tdsystem.h" #include "randomizer.icc"
```

Classes

- class Randomizer

  Randomizer.

7.85 resourceutilizationpoint.cc File Reference

```
#include "tdsystem.h"    #include "rtppacket.h"    #include
"resourceutilizationpoint.h"
```

Functions

- std::ostream & operator<< (std::ostream &os, const ResourceUtilizationPoint &rup)

7.85.1 Function Documentation

7.85.1.1 std::ostream& operator<< (std::ostream & *os,* const ResourceUtilizationPoint & *rup*)

Output operator.

7.86 resourceutilizationpoint.h File Reference

```
#include "tdsystem.h"    #include "rtppacket.h"    #include
"bandwidthinfo.h" #include "trafficclassvalues.h" #include
"resourceutilizationpoint.icc"
```

Classes

- struct LayerClassMappingPossibility

  Layer Class Mapping Possibility.
- struct LayerClassMapping

  Layer Class Mapping.
- class ResourceUtilizationPoint

  Resource Utilization Point.

Functions

- std::ostream & operator<< (std::ostream &os, const ResourceUtilizationPoint &rup)

7.86.1 Function Documentation

7.86.1.1 std::ostream& operator<< (std::ostream & *os,* const ResourceUtilizationPoint & *rup*)

Output operator.

7.87 ringbuffer.cc File Reference

```
#include "tdsystem.h" #include "ringbuffer.h"
```

7.88 ringbuffer.h File Reference

```
#include "tdsystem.h"   #include "condition.h"   #include
"ringbuffer.icc"
```

Classes

- class RingBuffer

  Ring Buffer.

7.89 roundtriptimepinger.cc File Reference

```
#include "tdsystem.h"     #include "roundtriptimepinger.-
h"   #include "tdsocket.h"   #include "internetaddress.h"
#include "breakdetector.h" #include "timedthread.h" #include
"rtppacket.h" #include "strings.h" #include "trafficclassvalues.-
h"
```

Classes

- struct icmp_filter

Defines

- #define ICMP_FILTER 1

Functions

- std::ostream & operator<< (std::ostream &os, RoundTripTimePinger &pinger)

7.89.1 Define Documentation

7.89.1.1 #define ICMP_FILTER 1

7.89.2 Function Documentation

7.89.2.1 std::ostream& operator<< (std::ostream & *os,* RoundTripTimePinger & *pinger*)

Friend output operator.

7.90 roundtriptimepinger.h File Reference

```
#include "tdsystem.h" #include "tdsocket.h" #include "internetaddress.-
h" #include "timedthread.h" #include "rtppacket.h" #include
"pingerhost.h"   #include "randomizer.h"   #include <set>
#include <algorithm> #include <fstream> #include <netinet/ip.-
h> #include <netinet/ip_icmp.h> #include <netinet/icmp6.-
h>  #include <sys/time.h>  #include "roundtriptimepinger.-
icc"
```

Classes

- class RoundTripTimePinger

  Round Trip Time Pinger.
- struct RoundTripTimePinger::Ping4Packet
- struct RoundTripTimePinger::Ping6Packet

7.91 rtcpabstractserver.cc File Reference

```
#include "tdsystem.h" #include "string.h" #include "rtcpabstractserver.-
h"
```

7.92 rtcpabstractserver.h File Reference

```
#include "tdsystem.h"  #include "timedthread.h"  #include
"rtcppacket.h" #include "internetflow.h" #include "sourcestateinfo.-
h" #include <map> #include "rtcpabstractserver.icc"
```

Classes

- class RTCPAbstractServer

  RTCP abstract server.

- struct RTCPAbstractServer::Client

7.93 rtcppacket.cc File Reference

```
#include "tdsystem.h" #include "rtcppacket.h"
```

7.94 rtcppacket.h File Reference

```
#include "tdsystem.h"   #include "rtppacket.h"   #include
"rtcppacket.icc"
```

Classes

- struct RTCPCommonHeader

  RTCP Common Header.
- struct RTCPSenderInfoBlock

  RTCP Sender Info Block.
- struct RTCPReceptionReportBlock

  RTCP Reception Report Block.
- struct RTCPReport

  RTCP Report.
- struct RTCPSenderReport

  RTCP Sender Report.
- struct RTCPReceiverReport

  RTCP Sender Report.
- struct RTCPSourceDescriptionItem

  RTCP Source Description Item.
- struct RTCPSourceDescriptionChunk

  RTCP Source Description Chunk.
- struct RTCPSourceDescription

  RTCP Source Description (SDES)
- struct RTCPBye

  RTCP BYE Message.
- struct RTCPApp

  RTCP APP Message.

Enumerations

- enum RTCP_Type { RTCP_SR = 200, RTCP_RR = 201, RTCP_SDES = 202, RTCP_BYE = 203, RTCP_APP = 204 }

- enum RTCP_SDES_Type { RTCP_SDES_END = 0, RTCP_SDES_CNAME = 1, RTCP_SDES_NAME = 2, RTCP_SDES_EMAIL = 3, RTCP_SDES_PHONE = 4, RTCP_SDES_LOC = 5, RTCP_SDES_TOOL = 6, RTCP_SDES_NOTE = 7, RTCP_SDES_PRIV = 8 }

Functions

- struct RTCPCommonHeader __attribute__ ((packed))
- RTCPCommonHeader ()
- card8 getVersion () const
- card8 getPadding () const
- card8 getCount () const
- card8 getPacketType () const
- card16 getLength () const
- void setVersion (const card8 version)
- void setPadding (const card8 padding)
- void setCount (const card8 count)
- void setPacketType (const card8 packetType)
- void setLength (const card16 length)
- RTCPSenderInfoBlock ()
- card64 getNTPTimeStamp () const
- card32 getRTPTimeStamp () const
- card32 getPacketsSent () const
- card32 getOctetsSent () const
- void setNTPTimeStamp (const card64 timeStamp)
- void setRTPTimeStamp (const card32 timeStamp)
- void setPacketsSent (const card32 packets)
- void setOctetsSent (const card32 octets)
- RTCPReceptionReportBlock ()
- RTCPReceptionReportBlock (const card32 ssrc)
- void init (const card32 ssrc)
- card32 getSSRC () const
- double getFractionLost () const
- card32 getPacketsLost () const
- card32 getLastSeqNum () const
- card32 getJitter () const
- card32 getLSR () const
- card32 getDLSR () const
- void setSSRC (card32 ssrc)
- void setFractionLost (const double fraction)
- void setPacketsLost (const card32 packetsLost)
- void setLastSeqNum (const card32 lastSeq)
- void setJitter (const card32 jitter)
- void setLSR (const card32 lsr)
- void setDLSR (const card32 dlsr)
- RTCPReport ()

- RTCPSenderReport ()
- RTCPSenderReport (const card32 syncSource, const card8 count=0)
- void init (const card32 syncSource, const card8 count=0)
- RTCPReceiverReport ()
- RTCPReceiverReport (const card32 syncSource, const card8 count=0)
- RTCPSourceDescription ()
- RTCPSourceDescription (const card8 count)
- void init (const card8 count)
- RTCPBye ()
- RTCPBye (const card8 count)
- card32 getSource (const cardinal index) const
- void setSource (const cardinal index, const card32 source)
- RTCPApp ()
- RTCPApp (const card8 subtype)
- card32 getSource () const
- char * getName ()
- char * getData ()
- void setSource (const card32 source)
- void setName (const char *name)

Variables

- card8 V
- card8 P
- card8 C
- card8 PT
- card16 Length
- card32 NTP_MostSignificant
- card32 NTP_LeastSignificant
- card32 RTPTimeStamp
- card32 PacketsSent
- card32 OctetsSent
- card32 SSRC
- card32 Fraction
- card32 Lost
- card32 LastSeq
- card32 Jitter
- card32 LSR
- card32 DLSR
- RTCPReceptionReportBlock rr []
- card8 Type
- char Data []
- card32 SRC
- RTCPSourceDescriptionItem Item []
- RTCPSourceDescriptionChunk Chunk [1]
- card32 Source []
- char Name [4]

7.94.1 Enumeration Type Documentation

7.94.1.1 enum RTCP_SDES_Type

Definition of RTCP SDES message types.

Enumerator:

RTCP_SDES_END

RTCP_SDES_CNAME

RTCP_SDES_NAME

RTCP_SDES_EMAIL

RTCP_SDES_PHONE

RTCP_SDES_LOC

RTCP_SDES_TOOL

RTCP_SDES_NOTE

RTCP_SDES_PRIV

7.94.1.2 enum RTCP_Type

Definition of RTCP message types.

Enumerator:

RTCP_SR

RTCP_RR

RTCP_SDES

RTCP_BYE

RTCP_APP

7.94.2 Function Documentation

7.94.2.1 struct RTCPCommonHeader __attribute__ ((packed))

7.94.2.2 card8 __attribute__::getCount ( ) const [inline]

Get count.

Returns

RTCP count.

7.94.2.3 char* __attribute__::getData ( ) [inline]

Get pointer to data field.

Returns

Pointer to data field.

7.94.2.4 card32 __attribute__::getDLSR ( ) const [inline]

Get DLSR.

Returns

DLSR.

7.94.2.5 double __attribute__::getFractionLost ( ) const [inline]

Get fraction lost.

Returns

Fraction lost.

7.94.2.6 card32 __attribute__::getJitter ( ) const [inline]

Get jitter.

Returns

Jitter.

7.94.2.7 card32 __attribute__::getLastSeqNum ( ) const [inline]

Get last sequence number.

Returns

Last sequence number.

7.94.2.8 card16 __attribute__::getLength ( ) const [inline]

Get length.

Returns

RTCP Length.

7.94.2.9 card32 __attribute__::getLSR ( ) const [inline]

Get LSR.

Returns

LSR.

7.94.2.10 char* __attribute__::getName ( ) [inline]

Get pointer to name field.

Returns

Pointer to name field.

7.94.2.11 card64 __attribute__::getNTPTimeStamp ( ) const [inline]

Get NTP timestamp.

Returns

NTP timestamp.

7.94.2.12 card32 __attribute__::getOctetsSent ( ) const [inline]

Get octets sent.

Returns

Octets sent.

7.94.2.13 card32 __attribute__::getPacketsLost ( ) const [inline]

Get packets lost.

Returns

Packets lost.

7.94.2.14 card32 __attribute__::getPacketsSent ( ) const [inline]

Get packets sent.

Returns

Packets sent.

7.94.2.15 card8 __attribute__::getPacketType ( ) const [inline]

Get packet type.

Returns

RTCP packet type.

7.94.2.16 card8 __attribute__::getPadding ( ) const [inline]

Get padding.

Returns

RTCP padding.

7.94.2.17 card32 __attribute__::getRTPTimeStamp ( ) const [inline]

Get RTP time stamp.

Returns

RTP time stamp.

7.94.2.18 card32 __attribute__::getSource (const cardinal *index*) const [inline]

Get source at given index.

Parameters

index	Index.

Returns

Source.

7.94.2.19 card32 __attribute__::getSource ( ) const [inline]

Get source.

Returns

Source.

7.94.2.20 card32 getSSRC ( ) const [inline]

Get SSRC.

Returns

SSRC.

7.94.2.21 card8 __attribute__::getVersion ( ) const [inline]

Get version.

Returns

RTCP version.

7.94.2.22 void __attribute__::init (const card32 *ssrc*)

Initialize.

Parameters

ssrc	SSRC.

7.94.2.23 void init (const card32 *syncSource*, const card8 *count* = 0)

Initialize.

Parameters

syncSource	SSRC.
count	Count.

7.94.2.24 void init (const card8 *count*)

Initialize.

Parameters

Count	count.

7.94.2.25 __attribute__::RTCPApp ( )

Constructor.

7.94.2.26 __attribute__::RTCPApp (const card8 *subtype*)

Constructor.

Parameters

subtype	RTCP APP subtype.

7.94.2.27 __attribute__::RTCPBye ( )

Constructor.

7.94.2.28 __attribute__::RTCPBye (const card8 *count*)

Constructor.

Parameters

Count	count.

7.94.2.29 __attribute__::RTCPCommonHeader ( )

Constructor.

7.94.2.30 __attribute__::RTCPReceiverReport ( )

Constructor.

7.94.2.31 __attribute__::RTCPReceiverReport (const card32 *syncSource,* const card8 *count* = 0)

Constructor.

Parameters

syncSource	SSRC.
count	Count.

7.94.2.32 __attribute__::RTCPReceptionReportBlock ( )

Constructor.

7.94.2.33 __attribute__::RTCPReceptionReportBlock (const card32 *ssrc*)

Constructor.

Parameters

ssrc	SSRC.

7.94.2.34 __attribute__::RTCPReport ( )

Constructor.

7.94.2.35 __attribute__::RTCPSenderInfoBlock ( )

Constructor.

7.94.2.36 __attribute__::RTCPSenderReport ( )

Constructor.

7.94.2.37 __attribute__::RTCPSenderReport (const card32 *syncSource,* const card8 *count* = 0)

Constructor.

Parameters

syncSource	SSRC.
count	Count.

7.94.2.38 __attribute__::RTCPSourceDescription ( )

Constructor.

7.94.2.39 __attribute__::RTCPSourceDescription (const card8 *count*)

Constructor.

Parameters

Count	count.

7.94.2.40 void __attribute__::setCount (const card8 *count*) [inline]

Set count.

Parameters

count	RTCP count.

7.94.2.41 void __attribute__::setDLSR (const card32 *dlsr*) [inline]

Set DLSR.

Parameters

dlsr	DLSR.

7.94.2.42 void __attribute__::setFractionLost (const double *fraction*) [inline]

Set fraction lost.

Parameters

fraction	Fraction lost.

7.94.2.43 void __attribute__::setJitter (const card32 *jitter*) [inline]

Set jitter.

Returns

jitter Jitter.

7.94.2.44 void __attribute__::setLastSeqNum (const card32 *lastSeq*) [inline]

Set last sequence number.

Parameters

lastSeq	Last sequence number.

7.94.2.45 void __attribute__::setLength (const card16 *length*) [inline]

Set length.

Parameters

length	RTCP Length.

7.94.2.46 void __attribute__::setLSR (const card32 *lsr*) [inline]

Set LSR.

Parameters

lsr	LSR.

7.94.2.47 void __attribute__::setName (const char * *name*) [inline]

Set name.

Returns

Pointer to name field.

7.94.2.48 void __attribute__::setNTPTimeStamp (const card64 *timeStamp*) [inline]

Set NTP timestamp.

Parameters

timeStamp	NTP timestamp.

7.94.2.49 void __attribute__::setOctetsSent (const card32 *octets*) [inline]

Set octets sent.

Parameters

octets	Octets sent.

7.94.2.50 void __attribute__::setPacketsLost (const card32 *packetsLost*) [inline]

Set packets lost.

Parameters

packetsLost	Packets lost.

7.94.2.51 void __attribute__::setPacketsSent (const card32 *packets*) [inline]

Set packets sent.

Parameters

packets	Packets sent.

7.94.2.52 void __attribute__::setPacketType (const card8 *packetType*) [inline]

Set packetType.

Parameters

packetType	RTCP packet Type.

7.94.2.53 void __attribute__::setPadding (const card8 *padding*) [inline]

Set padding.

Parameters

padding	RTCP padding.

7.94.2.54 void __attribute__::setRTPTimeStamp (const card32 *timeStamp*) [inline]

Set RTP time stamp.

Parameters

timeStamp	RTP timestamp.

7.94.2.55 void __attribute__::setSource (const cardinal *index,* const card32 *source*) [inline]

Set source at given index.

Parameters

index	Index.
source	Source.

7.94.2.56 void __attribute__::setSource (const card32 *source*) [inline]

Set source.

Parameters

source	Source.

7.94.2.57 void setSSRC (card32 *ssrc*) [inline]

Set SSRC.

Parameters

ssrc	SSRC.

7.94.2.58 void __attribute__::setVersion (const card8 *version*) [inline]

Set version.

Parameters

version	RTCP version.

7.94.3 Variable Documentation

7.94.3.1 card8 C

7.94.3.2 RTCPSourceDescriptionChunk Chunk[1]

Array of SDES chunks.

7.94.3.3 char Data[]

Item data.

7.94.3.4 card32 DLSR

7.94.3.5 card32 Fraction

7.94.3.6 RTCPSourceDescriptionItem Item[ ]

Array of SDES items.

7.94.3.7 card32 Jitter

7.94.3.8 card32 LastSeq

7.94.3.9 card8 Length

Length in bytes.

7.94.3.10 card32 Lost

7.94.3.11 card32 LSR

7.94.3.12 char Name[4]

7.94.3.13 card32 NTP_LeastSignificant

7.94.3.14 card32 NTP_MostSignificant

7.94.3.15 card32 OctetsSent

7.94.3.16 card8 P

7.94.3.17 card32 PacketsSent

7.94.3.18 card8 PT

7.94.3.19 RTCPReceptionReportBlock rr

Array of RTCPReceptionReportBlocks

7.94.3.20 card32 RTPTimeStamp

7.94.3.21 card32 Source

7.94.3.22 card32 SRC

SSRC/CSRC Identifier of source.

7.94.3.23 card32 SSRC

7.94.3.24 card8 Type

Item type (RTCP_SDES_...).

7.94.3.25 card8 V

7.95 rtcpreceiver.cc File Reference

```
#include "tdsystem.h"  #include "rtcpreceiver.h"  #include
"internetflow.h" #include "tools.h"
```

Defines

- #define DEBUG

7.95.1 Define Documentation

7.95.1.1 #define DEBUG

7.96 rtcpreceiver.h File Reference

```
#include "tdsystem.h" #include "tdsocket.h" #include "thread.-
h" #include "rtcppacket.h" #include "rtcpabstractserver.-
h"
```

Classes

- class RTCPReceiver
  RTCP Receiver.

Variables

- RTCPReceiver __attribute__

7.96.1 Variable Documentation

7.96.1.1 RTCPApp RTCPCommonHeader __attribute__

7.97 rtcpsender.cc File Reference

```
#include "tdsystem.h"   #include "tdmessage.h"   #include
"rtcpsender.h"
```

7.98 rtcpsender.h File Reference

```
#include "tdsystem.h" #include "tdsocket.h" #include "timedthread.-
h" #include "rtcppacket.h" #include "rtpreceiver.h" #include
"randomizer.h" #include <map>
```

Classes

- class RTCPSender

  RTCP Sender.

7.99 rtpa-client.cc File Reference

```
#include "multiaudiowriter.h"   #include "audiodevice.h" ×
#include "audiodebug.h"  #include "audionull.h"  #include
"tdsocket.h" #include "audioclient.h" #include "breakdetector.-
h" #include "audioquality.h" #include "strings.h" #include
<assert.h> #include <fstream>
```

Functions

- void cleanUp (const cardinal exitCode=0)
- void initGNUplot (const char ∗prefix, const char ∗info, const cardinal layers, const String &address)
- int main (int argc, char ∗argv[])

Variables

- AudioWriterInterface ∗ audioOutput = NULL
- AudioClient ∗ client = NULL
- std::ofstream ∗ gpScript = NULL
- std::ofstream ∗ gpData = NULL

7.99.1 Function Documentation

7.99.1.1 void cleanUp (const cardinal *exitCode* = 0)

7.99.1.2 void initGNUplot (const char ∗ *prefix,* const char ∗ *info,* const cardinal *layers,* const String & *address*)

7.99.1.3 int main (int *argc,* char ∗ *argv[]*)

7.99.2 Variable Documentation

7.99.2.1 AudioWriterInterface* audioOutput = NULL

7.99.2.2 AudioClient* client = NULL

7.99.2.3 std::ofstream* gpData = NULL

7.99.2.4 std::ofstream* gpScript = NULL

7.100 rtpa-qclient.cc File Reference

```
#include "tdsystem.h"  #include "rtpa-qclient.h"  #include
"rtpa-qclient_moc.cc"  #include "audiodevice.h"  #include
"audiodebug.h" #include "audionull.h" #include "spectrumanalyzer.-
h"   #include "multiaudiowriter.h"   #include "tdsocket.-
h"  #include "tools.h"  #include "audioclient.h"  #include
"trafficclassvalues.h" #include <qapplication.h> #include
<qlayout.h> #include <qpushbutton.h> #include <qscrollbar.-
h> #include <qlcdnumber.h> #include <qlineedit.h> #include
<qbuttongroup.h> #include <qcheckbox.h> #include <qcombobox.-
h> #include <qradiobutton.h> #include <qframe.h> #include
<qgroupbox.h>  #include <qlabel.h>  #include <qtimer.h> ×
#include <qmessagebox.h>  #include <qmenubar.h>  #include
<qmenu.h> #include <fstream> #include <assert.h>
```

Functions

- int main (int argc, char *argv[])

7.100.1 Function Documentation

7.100.1.1 int main (int *argc*, char * *argv[]*)

7.101 rtpa-qclient.h File Reference

```
#include "audiowriterinterface.h" #include "spectrumanalyzer.-
h"  #include "audiomixer.h"  #include "tools.h"  #include
"strings.h" #include "audioclient.h" #include <qapplication.-
h> #include <qlayout.h> #include <qpushbutton.h> #include
<qscrollbar.h> #include <qlineedit.h> #include <qbuttongroup.-
h> #include <qcheckbox.h> #include <qcombobox.h> #include
<qradiobutton.h> #include <qframe.h> #include <qgroupbox.-
h>  #include <qlabel.h>  #include <qlcdnumber.h>  #include
<qwhatsthis.h> #include <qmainwindow.h> #include <qlist.-
h> #include "qspectrumanalyzer.h" #include "qaudiomixer.-
h" #include "qinfotabwidget.h"
```

Classes

- class QClient

  QClient.

Variables

- const InfoEntry InfoEntries1 []
- const InfoTable InfoTable1
- const InfoEntry InfoEntries2 []
- const InfoTable InfoTable2

7.101.1 Variable Documentation

7.101.1.1 const InfoEntry InfoEntries1[]

Initial value:

```
{
  {"SA",    "Server Address",   "This are the IPv4 or IPv6 address and port
      number of the audio server."},
  {"TF",    "TOS/Flow Label",   "This are the TOS/traffic class values for
      each layer and the flow label "
                                "(IPv6 only) of the received packets."},
  {"SSSRC", "Server SSRC",      "This is the audio server's RTP SSRC. It is a
      32-bit random number."},
  {"CA",    "Client Address",   "This is the IPv4 or IPv6 address and port
      number of the audio client."},
  {"CSSRC", "Client SSRC",      "This is the audio clients's RTP SSRC. It is a
      32-bit random number."},
  {"BR",    "Bytes Received",   "This is a counter for the number of bytes
      received from server "
                                "(IP/UDP/RTP/RTP Audio headers + payload)."},
  {"PR",    "Packets Received", "This is a counter for the number of packets
      received from server. "
                                "The bytes per second value is the value for
      the quality received "
                                "from server (IP/UDP/RTP/RTP Audio headers +
      payload)."},
  {"PL",    "Packets Lost",     "This is a counter for the number of packets
      lost during transmission."
                                "The loss fraction shows the fraction of
      packets lost during the last "
                                "RTCP report interval in each layer."},
  {"IJ", "Interarrival Jitter", "This is the interarrival jitter: An estimate
      of the statistical variance of "
                                "the RTP data packet interarrival time,
      measured in milliseconds.\n\n"
                                "Definition:\n"
                                "Let Si, Sj be the RTP timestamps of packets
      i, j.\n"
                                "Let Ri, Rj be the arrival timestamps.\n"
                                "Dij := (Rj - Sj) - (Ri - Si).\n"
                                "Jitter := Jitter + (1.0/16.0) * abs(Dij).\n\n
     "
```

```
                                    "See RFC 1889, Page 25-26 for more details."},
   {"Q", "Quality",  "This is the audio quality received from server: Sampling
       rate, bits and channels."},
   {"E", "Encoding", "This is the name of the audio encoding format received
       from server."},
}
```

Transmission status info table #1 entries.

7.101.1.2 const InfoEntry InfoEntries2[]

Initial value:

```
 {
   {"LSA", "Source",           "This is the current layer's source address and
       port number."},
   {"LTF", "TOS/Flow Label",   "This are the current layer's traffic class and
       flow label (IPv6 only)."},
   {"CA",  "Destination",      "This is the current layer's destination address
       and port number."},
   {"LPR", "Packets Received", "This is the number of packets received in this
       layer."},
   {"LPL", "Packets Lost",     "This is the number of packets lost in this
       layer."},
   {"LFL", "Fraction Lost",    "This is the fraction of packets lost in this
       layer."},
   {"LBR", "Bytes Received",   "This is the sum of bytes received in this
       layer."},
   {"LIJ", "Interarrival Jitter", "This is the interarrival jitter of this
       layer."},
   {"Q", "Quality",  "This is the audio quality received from server: Sampling
       rate, bits and channels."},
   {"E", "Encoding", "This is the name of the audio encoding format received
       from server."},
}
```

Transmission status info table #1 entries.

7.101.1.3 const InfoTable InfoTable1

Initial value:

```
{
   sizeof(InfoEntries1) / sizeof(InfoEntry),
   (const InfoEntry*)&InfoEntries1
}
```

Transmission status info table #1.

7.101.1.4 const InfoTable InfoTable2

Initial value:

```
{
```

```
   sizeof(InfoEntries2) / sizeof(InfoEntry),
   (const InfoEntry*)&InfoEntries2
}
```

Transmission status info table #2.

7.102 rtpa-qclient_moc.cc File Reference

```
#include "rtpa-qclient.h"
```

Variables

- static QT_BEGIN_MOC_NAMESPACE const uint qt_meta_data_QClient []
- static const char qt_meta_stringdata_QClient []

7.102.1 Variable Documentation

7.102.1.1 QT_BEGIN_MOC_NAMESPACE const uint qt_meta_data_QClient[] [static]

7.102.1.2 const char qt_meta_stringdata_QClient[] [static]

Initial value:

```
 {
    "QClient\0\0play()\0stop()\0information()\0"
    "whatsThis()\0on\0pause(bool)\0togglePause()\0"
    "selected\0toggleAddressResolution(bool)\0"
    "spectrumAnalyzer()\0audioMixer()\0"
    "closeSpectrumAnalyzer()\0closeAudioMixer()\0"
    "quit()\0value\0position(int)\0index\0"
    "setSamplingRate(int)\0stereo\0"
    "setChannels(bool)\0setBits(int)\0"
    "setEncoding(int)\0action\0"
    "locationSelected(QAction*)\0loadBookmarks()\0"
    "clearBookmarks()\0saveBookmarks()\0"
    "timerEvent()\0"
}
```

7.103 rtpa-server.cc File Reference

```
#include "tdsystem.h" #include "tdsocket.h" #include "rtpsender.-
h"   #include "rtcppacket.h"   #include "rtcpreceiver.h" ×
#include "rtcpabstractserver.h" #include "audioclientapppacket.-
h"  #include "audioserver.h"  #include "tools.h"  #include
"breakdetector.h"  #include "bandwidthmanager.h"  #include
"servicelevelagreement.h" #include <fstream>
```

Defines

- #define WITH_QOSMGR

Functions

- void cleanUp (const cardinal exitCode=0)
- void initAll (const char *directory, const card16 port, const card64 timeout, const cardinal maxPacketSize, const bool lossScalability, const bool useSCTP)
- int main (int argc, char *argv[])

Variables

- static Socket * rtcpServerSocket = NULL
- static RTCPReceiver * rtcpReceiver = NULL
- static AudioServer * server = NULL
- static BandwidthManager * qosManager = NULL
- static ServiceLevelAgreement * sla = NULL
- static Socket * pingSocket4 = NULL
- static Socket * pingSocket6 = NULL
- static RoundTripTimePinger * pinger = NULL
- static std::ofstream * logStream = NULL

7.103.1 Define Documentation

7.103.1.1 #define WITH_QOSMGR

7.103.2 Function Documentation

7.103.2.1 void cleanUp (const cardinal *exitCode* = `0`)

7.103.2.2 void initAll (const char * *directory*, const card16 *port*, const card64 *timeout*, const cardinal *maxPacketSize*, const bool *lossScalability*, const bool *useSCTP*)

7.103.2.3 int main (int *argc*, char * *argv[]*)

7.103.3 Variable Documentation

7.103.3.1 std::ofstream* logStream = NULL [static]

7.103.3.2 RoundTripTimePinger* pinger = NULL [static]

7.103.3.3 Socket* pingSocket4 = NULL [static]

7.103.3.4 Socket* pingSocket6 = NULL [static]

7.103.3.5 BandwidthManager* qosManager = NULL [static]

7.103.3.6 RTCPReceiver* rtcpReceiver = NULL [static]

7.103.3.7 Socket* rtcpServerSocket = NULL [static]

7.103.3.8 AudioServer* server = NULL [static]

7.103.3.9 ServiceLevelAgreement* sla = NULL [static]

7.104 rtpa-vclient.cc File Reference

```
#include "tdsystem.h"   #include "audionull.h"   #include
"audioclient.h" #include "randomizer.h" #include "thread.-
h"   #include "mediainfo.h"   #include "breakdetector.h" ×
#include <sys/types.h> #include <signal.h>
```

Classes

- class VerificationClientThread

Functions

- void validatePr (double &p)
- int main (int argc, char **argv)

Variables

- const cardinal DefaultPause = 500
- const cardinal DefaultThreads = 12
- const char * DefaultURL = "rtpa+udp://localhost:7500/Test%u.list"
- const cardinal DefaultMediaCount = 4

7.104.1 Function Documentation

7.104.1.1 int main (int *argc,* char ** *argv*)

7.104.1.2 void validatePr (double & *p*) [inline]

7.104.2 Variable Documentation

7.104.2.1 const cardinal DefaultMediaCount = 4

7.104.2.2 const cardinal DefaultPause = 500

7.104.2.3 const cardinal DefaultThreads = 12

7.104.2.4 const char* DefaultURL = "rtpa+udp://localhost:7500/Test%u.list"

7.105 rtppacket.cc File Reference

`#include "tdsystem.h" #include "rtppacket.h"`

Functions

- std::ostream & operator<< (std::ostream &os, const RTPPacket &packet)

7.105.1 Function Documentation

7.105.1.1 std::ostream& operator<< (std::ostream & *os,* const RTPPacket & *packet*)

Output operator.

7.106 rtppacket.h File Reference

`#include "tdsystem.h" #include "tdsocket.h" #include "rtppacket.-icc"`

Classes

- struct RTPPacket

  RTP Packet.

Namespaces

- namespace RTPConstants

Functions

- struct RTPPacket __attribute__ ((packed))
- RTPPacket ()
- card8 getVersion () const
- card8 getPadding () const
- card8 getExtension () const
- card8 getCSRCCount () const
- bool getMarker () const
- card8 getPayloadType () const

- card16 getSequenceNumber () const
- card32 getTimeStamp () const
- card32 getSSRC () const
- card32 getCSRC (cardinal index) const
- cardinal calculateHeaderSize () const
- char * getPayloadData () const
- cardinal getMaxPayloadSize () const
- void setVersion (const card8 version)
- void setPadding (const card8 padding)
- void setExtension (const card8 extension)
- void setCSRCCount (const card8 count)
- void setMarker (const bool marker)
- void setPayloadType (const card8 payloadType)
- void setSequenceNumber (const card16 sequenceNumber)
- void setTimeStamp (const card32 timeStamp)
- void setSSRC (const card32 ssrc)
- void setCSRC (const cardinal index, const card32 csrc)
- std::ostream & operator<< (std::ostream &os, const RTPPacket &packet)

Variables

- const cardinal RTPConstants::RTPMaxPayloadLimit = 8192
- const cardinal RTPConstants::RTPDefaultMaxPayload = 1376
- const cardinal RTPConstants::RTPDefaultHeaderSize = 12
- const card8 RTPConstants::RTPVersion = 2
- const double RTPConstants::RTPMicroSecondsPerTimeStamp = 1000.0 / 16.0
- const cardinal RTPConstants::RTPMaxQualityLayers = 16
- card8 V
- card8 P
- card8 X
- card8 CC
- card8 M
- card8 PT
- card16 SequenceNumber
- card32 TimeStamp
- card32 SSRC
- card32 CSRC [16]
- char Data [RTPConstants::RTPMaxPayloadLimit]

7.106.1 Function Documentation

7.106.1.1 struct RTPPacket __attribute__ ((packed))

7.106.1.2 cardinal __attribute__::calculateHeaderSize ( ) const [inline]

Calculate header size.

Returns

Header size.

7.106.1.3 card32 __attribute__::getCSRC (cardinal *index*) const [inline]

Get CSRC at given index.

Parameters

index	Index.

Returns

RTP CSRC.

7.106.1.4 card8 __attribute__::getCSRCCount ( ) const [inline]

Get CSRC count.

Returns

RTP CSRC count.

7.106.1.5 card8 __attribute__::getExtension ( ) const [inline]

Get extension.

Returns

RTP Extension.

7.106.1.6 bool __attribute__::getMarker ( ) const [inline]

Get marker.

Returns

RTP Marker.

7.106.1.7 cardinal __attribute__::getMaxPayloadSize ( ) const [inline]

Get maximum payload size.

Returns

Maximum payload size.

7.106.1.8 card8 __attribute__::getPadding ( ) const [inline]

Get padding.

Returns

RTP Padding.

7.106.1.9 char* __attribute__::getPayloadData ( ) const [inline]

Get pointer to payload data.

Returns

pointer to payload data.

7.106.1.10 card8 __attribute__::getPayloadType ( ) const [inline]

Get payload type.

Returns

RTP Payload type.

7.106.1.11 card16 __attribute__::getSequenceNumber ( ) const [inline]

Get sequence number.

Returns

RTP Sequence number.

7.106.1.12 card32 __attribute__::getSSRC ( ) const [inline]

Get SSRC.

Returns

RTP SSRC.

7.106.1.13 card32 __attribute__::getTimeStamp ( ) const [inline]

Get time stamp.

Returns

RTP Time stamp.

7.106.1.14 card8 __attribute__::getVersion ( ) const [inline]

Get version.

Returns

RTP Version.

7.106.1.15 friend std::ostream& __attribute__::operator<< (std::ostream & *os*, const RTPPacket & *packet*)

Output operator.

7.106.1.16 __attribute__::RTPPacket ( )

Constructor.

7.106.1.17 void __attribute__::setCSRC (const cardinal *index*, const card32 *csrc*) [inline]

Set CSRC at given index.

Parameters

index	Index.
csrc	CSRC.

7.106.1.18 void __attribute__::setCSRCCount (const card8 *count*) [inline]

Set CSRC count.

Parameters

count	RTP CSRC count.

7.106.1.19 void __attribute__::setExtension (const card8 *extension*) [inline]

Set extension.

Parameters

extension	RTP Extension.

7.106.1.20 void __attribute__::setMarker (const bool *marker*) [inline]

Set marker.

Parameters

marker	RTP Marker.

7.106.1.21 void __attribute__::setPadding (const card8 *padding*) [inline]

Set padding.

Parameters

padding	RTP Padding.

7.106.1.22 void __attribute__::setPayloadType (const card8 *payloadType*) [inline]

Set payload type.

Parameters

payloadType	RTP Payload type.

7.106.1.23 void __attribute__::setSequenceNumber (const card16 *sequenceNumber*) [inline]

Set sequence number.

Parameters

sequence-Number	RTP Sequence number.

7.106.1.24 void __attribute__::setSSRC (const card32 *ssrc*) [inline]

Set SSRC.

Parameters

ssrc	RTP SSRC.

7.106.1.25 void __attribute__::setTimeStamp (const card32 *timeStamp*) [inline]

Set time stamp.

Parameters

timeStamp	RTP timeStamp.

7.106.1.26 void __attribute__::setVersion (const card8 *version*) [inline]

Set version.

Parameters

version	RTP Version.

7.106.2 Variable Documentation

7.106.2.1 card8 CC

7.106.2.2 card32 CSRC[16]

7.106.2.3 char Data[RTPConstants::RTPMaxPayloadLimit]

7.106.2.4 card8 M

7.106.2.5 card8 P

7.106.2.6 card8 PT

7.106.2.7 card16 SequenceNumber

7.106.2.8 card32 SSRC

7.106.2.9 card32 TimeStamp

7.106.2.10 card8 V

7.106.2.11 card8 X

7.107 rtpreceiver.cc File Reference

```
#include "tdsystem.h"  #include "rtpreceiver.h"  #include
"rtcppacket.h" #include "internetflow.h" #include "randomizer.-
h"
```

Defines

- #define DEBUG

7.107.1 Define Documentation

7.107.1.1 #define DEBUG

7.108 rtpreceiver.h File Reference

```
#include "tdsystem.h" #include "thread.h" #include "tdsocket.-
h" #include "rtppacket.h" #include "decoderinterface.h" ×
#include "sourcestateinfo.h" #include "internetflow.h" ×
#include "rtpreceiver.icc"
```

Classes

- class RTPReceiver

  RTP Receiver.

Variables

- RTPReceiver __attribute__

7.108.1 Variable Documentation

7.108.1.1 RTPReceiver __attribute__

7.109 rtpsender.cc File Reference

```
#include "tdsystem.h" #include "tdmessage.h" #include
"rtpsender.h" #include "rtcpsender.h" #include "randomizer.-
h" #include <signal.h>
```

Defines

- #define DEBUG

7.109.1 Define Documentation

7.109.1.1 #define DEBUG

7.110 rtpsender.h File Reference

```
#include "tdsystem.h"  #include "timedthread.h"  #include
"tdsocket.h" #include "rtppacket.h" #include "encoderinterface.-
h" #include "trafficshaper.h" #include "abstractqosdescription.-
h" #include "qosmanagerinterface.h" #include "rtpsender.-
icc"
```

Classes

- class RTPSender

  RTP Sender.

7.111 s2.cc File Reference

```
#include "tdsystem.h" #include "tdsocket.h" #include "rtpsender.-
h"  #include "rtcppacket.h"  #include "rtcpreceiver.h" ×
#include "rtcpabstractserver.h" #include "audioclientapppacket.-
h" #include "tools.h" #include "breakdetector.h" #include
"simpleaudioencoder.h"  #include "advancedaudioencoder.h"
#include "multiaudioreader.h" #include "audioencoderrepository.-
h"
```

Classes

- class MessageQueue< T >
- struct MessageQueue< T >::Message
- class AbstractMediaServentRequest
- class AudioServentRequest
- struct MediaServentLayerReport
- struct MediaServentReport
- class MediaServent
- class AbstractMediaServer
- class AlphaServent
- class AlphaServer
- class RTPAdaptionLayer
- class TestReceiver

  RTCP Receiver.

Defines

- #define SCTP_MAXADDRESSES 20
- #define VERBOSE
- #define DEBUG

Enumerations

- enum DeleteReason { DeleteReason_UserBye = 0, DeleteReason_Timeout = 1, DeleteReason_Shutdown = 2, DeleteReason_Error = 3 }
- enum ShutdownReason { SR_NoShutdown = 0, SR_UserShutdown = 1, SR_ServerShutdown = 2, SR_Timeout = 3, SR_Error = 255 }
- enum ServentQueueErrors { SQE_Okay = 0, SQE_NotForMe = 1, SQE_Invalid = 2, SQE_AuthFailed = 3, SQE_NoMemory = 255 }

Functions

- template<class T >
  ostream & operator<< (ostream &os, MessageQueue< T > &queue)
- void cleanUp (const cardinal exitCode=0)
- void initAll (const char *directory, SocketAddress **localAddressArray, const cardinal localAddresses, const card16 port, const card64 timeout, const cardinal maxPacketSize, const bool lossScalability, const bool useSCTP)
- int main (int argc, char *argv[])

Variables

- const cardinal MaxMediaServentLayers = 16
- Socket * rtcpServerSocket = NULL
- TestReceiver * rtcpReceiver = NULL
- RTPAdaptionLayer * adapt = NULL
- AlphaServer * server = NULL
- BandwidthManager * qosManager = NULL

7.111.1 Define Documentation

7.111.1.1 #define DEBUG

7.111.1.2 #define SCTP_MAXADDRESSES 20

7.111.1.3 #define VERBOSE

7.111.2 Enumeration Type Documentation

7.111.2.1 enum DeleteReason

Enumerator:

DeleteReason_UserBye

DeleteReason_Timeout

DeleteReason_Shutdown

DeleteReason_Error

7.111.2.2 enum ServentQueueErrors

Enumerator:

SQE_Okay

SQE_NotForMe

SQE_Invalid

SQE_AuthFailed

SQE_NoMemory

7.111.2.3 enum ShutdownReason

Enumerator:

SR_NoShutdown

SR_UserShutdown

SR_ServerShutdown

SR_Timeout

SR_Error

7.111.3 Function Documentation

7.111.3.1 void cleanUp (const cardinal *exitCode* = 0)

7.111.3.2 void initAll (const char * *directory*, SocketAddress ** *localAddressArray*, const cardinal *localAddresses*, const card16 *port*, const card64 *timeout*, const cardinal *maxPacketSize*, const bool *lossScalability*, const bool *useSCTP*)

7.111.3.3 int main (int *argc*, char * *argv[]*)

7.111.3.4 template<class T > ostream& operator<< (ostream & *os*, MessageQueue< T > & *queue*)

7.111.4 Variable Documentation

7.111.4.1 RTPAdaptionLayer* adapt = NULL

7.111.4.2 const cardinal MaxMediaServentLayers = 16

7.111.4.3 BandwidthManager* qosManager = NULL

7.111.4.4 TestReceiver* rtcpReceiver = NULL

7.111.4.5 Socket* rtcpServerSocket = NULL

7.111.4.6 AlphaServer* server = NULL

7.112 seqnumvalidator.cc File Reference

```
#include "tdsystem.h" #include "seqnumvalidator.h" #include
"rtppacket.h" #include "tools.h"
```

7.113 seqnumvalidator.h File Reference

```
#include "tdsystem.h" #include "seqnumvalidator.icc"
```

Classes

- class SeqNumValidator

  Sequence Number Validator.

7.114 servicelevelagreement.cc File Reference

```
#include "tdsystem.h" #include "servicelevelagreement.h"
```

Functions

- std::ostream & operator<< (std::ostream &os, const ServiceLevelAgreement sla)

7.114.1 Function Documentation

7.114.1.1 std::ostream& operator<< (std::ostream & *os,* const ServiceLevelAgreement *config*)

Output operator.

7.115 servicelevelagreement.h File Reference

```
#include "tdsystem.h" #include "bandwidthinfo.h" #include
"trafficclassvalues.h" #include "abstractlayerdescription.-
h" #include "servicelevelagreement.icc"
```

Classes

- struct DiffServClass

DiffServ Class.

- class ServiceLevelAgreement

  Trace Layer Configuration.

Functions

- std::ostream & operator<< (std::ostream &os, const ServiceLevelAgreement config)

7.115.1 Function Documentation

7.115.1.1 std::ostream& operator<< (std::ostream & *os,* const ServiceLevelAgreement *config*)

Output operator.

7.116 sessiondescription.h File Reference

```
#include "tdsystem.h"     #include "streamdescription.h" ×
#include <map>
```

Classes

- struct SessionDescription

  Session Description.

7.117 simpleaudiodecoder.cc File Reference

```
#include "tdsystem.h"    #include "simpleaudiodecoder.h" ×
#include "simpleaudiopacket.h" #include "seqnumvalidator.-
h" #include "tools.h"
```

7.118 simpleaudiodecoder.h File Reference

```
#include "tdsystem.h" #include "synchronizable.h" #include
"audiowriterinterface.h" #include "audiodecoderinterface.-
h" #include "seqnumvalidator.h" #include "audioquality.h"
```

Classes

- class SimpleAudioDecoder

  Simple Audio Decoder.

7.119 simpleaudioencoder.cc File Reference

```
#include "tdsystem.h"    #include "simpleaudioencoder.h" ×
#include "simpleaudiopacket.h" #include "tools.h" #include
"audioconverter.h"
```

7.120 simpleaudioencoder.h File Reference

```
#include "tdsystem.h"   #include "audioencoderinterface.h"
#include "audioreaderinterface.h" #include "audioquality.-
h"
```

Classes

- class SimpleAudioEncoder

  Simple Audio Encoder.

Defines

- #define SIMPLEAUDIOENCDOER_H

7.120.1 Define Documentation

7.120.1.1 #define SIMPLEAUDIOENCDOER_H

7.121 simpleaudiopacket.cc File Reference

```
#include "tdsystem.h"     #include "simpleaudiopacket.h" ×
#include "tools.h"
```

7.122 simpleaudiopacket.h File Reference

```
#include "tdsystem.h"    #include "mediainfo.h"    #include
"audioquality.h"
```

Classes

- struct SimpleAudioPacket

  Simple Audio Packet.

Enumerations

- enum SimpleAudioFlags { SAF_Data = 0, SAF_MediaInfo = 1 }

Functions

- struct SimpleAudioPacket __attribute__ ((packed))
- SimpleAudioPacket ()
- void translate ()
- void reset ()
- static AudioQuality calculateQualityForLimits (const AudioQualityInterface &userSetting, const AudioQualityInterface &inputQuality, const card64 totalByteRateLimit, const cardinal networkQualityDecrement, const cardinal headerSize, const cardinal maxPacketSize)
- static cardinal calculateFrameSize (const cardinal inputBytesPerSecond, const cardinal inputFrameSize)

Variables

- static const card16 SimpleAudioTypeID = 0x2960
- static const char SimpleAudioTypeName []
- static const card32 SimpleAudioFormatID = 0x74660000 | SimpleAudioTypeID
- static const cardinal SimpleAudioMediaInfoPacketsPerSecond = 1
- static const cardinal SimpleAudioFramesPerSecond = 15
- static const cardinal SimpleAudioFrameSize = 2352 * 5
- static const cardinal SimpleAudioMaxTransferDelay = 1500 * 16
- static const cardinal SimpleAudioQualityLevels = AudioQuality::QualityLevels
- card32 FormatID
- card16 SamplingRate
- card8 Channels
- card8 Bits
- card64 Position
- card64 MaxPosition
- card8 ErrorCode
- card8 Flags
- char Data []

7.122.1 Enumeration Type Documentation

7.122.1.1 enum SimpleAudioFlags

Emumeration of Flags.

Enumerator:

SAF_Data

SAF_MediaInfo

7.122.2 Function Documentation

7.122.2.1 struct SimpleAudioPacket __attribute__ ((packed))

7.122.2.2 static cardinal __attribute__::calculateFrameSize (const cardinal *inputBytesPerSecond,* const cardinal *inputFrameSize*) [static]

Calculate output frame size from given input bytes per second and input frame size.

Parameters

inputBytesPerSecond	Input source's bytes per second.
inputFrameSize	Input source's frame size.

Returns

The calculated frame size.

7.122.2.3 static AudioQuality __attribute__::calculateQualityForLimits (const AudioQualityInterface & *userSetting,* const AudioQualityInterface & *inputQuality,* const card64 *totalByteRateLimit,* const cardinal *networkQualityDecrement,* const cardinal *headerSize,* const cardinal *maxPacketSize*) [static]

Quality calculation for given user quality limited by input quality, byte rate and network quality decrement with given header size (eg. IP + UDP + RTP) and maximum packet size.

Parameters

userSetting	User's quality setting.
inputQuality	Input source's quality.
byteRateLimit	Byte rate limit.

networkQualityDecrement	Number of steps for decrement of user's quality.
headerSize	Header size (eg. IP + UDP + RTP). SimpleAudioPacket size is added automatically.
maxPacketSize	Maximum packet size.

Returns

The calculated quality.

7.122.2.4 void __attribute__::reset ( )

Reset report.

7.122.2.5 __attribute__::SimpleAudioPacket ( )

Constructor.

7.122.2.6 void __attribute__::translate ( )

Translate byte order.

7.122.3 Variable Documentation

7.122.3.1 card8 Bits

Number of audio bits.

7.122.3.2 card8 Channels

Number of audio channels.

7.122.3.3 char Data[ ]

Packet data.

7.122.3.4 card8 ErrorCode

Error code.

7.122.3.5 card8 Flags

Flags.

7.122.3.6 card32 FormatID

Packet format ID.

7.122.3.7 card64 MaxPosition

Maximum position in nanoseconds.

7.122.3.8 card64 Position

Current position in nanoseconds.

7.122.3.9 card16 SamplingRate

Audio sampling rate.

7.122.3.10 const card32 SimpleAudioFormatID = 0x74660000 | SimpleAudioTypeID [static]

Simple Audio Encoding package format ID.

7.122.3.11 const cardinal SimpleAudioFrameSize = 2352 * 5 [static]

Simple Audio frame size.

7.122.3.12 const cardinal SimpleAudioFramesPerSecond = 15 [static]

Simple Audio frames per second.

7.122.3.13 const cardinal SimpleAudioMaxTransferDelay = 1500 * 16 [static]

Simple Audio maximum transfer delay.

7.122.3.14 const cardinal SimpleAudioMediaInfoPacketsPerSecond = 1 [static]

Simple Audio MediaInfo packets per second.

7.122.3.15 const cardinal SimpleAudioQualityLevels = AudioQuality::QualityLevels [static]

Simple Audio number of quality levels.

7.122.3.16 const card16 SimpleAudioTypeID = 0x2960 [static]

Type ID for Simple Audio Encoding.

7.122.3.17 const char SimpleAudioTypeName[] [static]

Name for Simple Audio Encoding.

7.123 socketaddress.cc File Reference

```
#include "tdsystem.h" #include "socketaddress.h" #include
"internetaddress.h"   #include "unixaddress.h"   #include
"ext_socket.h" #include <sys/socket.h>
```

7.124 socketaddress.h File Reference

```
#include "tdsystem.h"   #include "tdstrings.h"   #include
<sys/socket.h> #include <sys/un.h> #include "socketaddress.-
icc"
```

Classes

- class SocketAddress

  Socket Address.

Functions

- std::ostream & operator<< (std::ostream &os, const SocketAddress &sa)

7.124.1 Function Documentation

7.124.1.1 std::ostream& operator<< (std::ostream & *os,* const SocketAddress & *sa*) [inline]

Output operator.

7.125 sourcestateinfo.cc File Reference

```
#include "sourcestateinfo.h"
```

Defines

- #define RTP_SEQ_MOD (1 << 16)
- #define RTP_MAX_DROPOUT 3000
- #define RTP_MAX_MISORDER 100
- #define RTP_MIN_SEQUENTIAL 2

7.125.1 Define Documentation

7.125.1.1 #define RTP_MAX_DROPOUT 3000

7.125.1.2 #define RTP_MAX_MISORDER 100

7.125.1.3 #define RTP_MIN_SEQUENTIAL 2

7.125.1.4 #define RTP_SEQ_MOD (1 << 16)

7.126 sourcestateinfo.h File Reference

```
#include "tdsystem.h" #include "synchronizable.h" #include
"seqnumvalidator.h" #include "sourcestateinfo.icc"
```

Classes

- class SourceStateInfo

  Source State Info.

7.127 spectrumanalyzer.cc File Reference

```
#include "tdsystem.h" #include "spectrumanalyzer.h" #include
"fft.h"   #include "tools.h"   #include "audioquality.h" ×
#include "audioconverter.h"  #include <iostream>  #include
<sys/time.h>
```

7.128 spectrumanalyzer.h File Reference

```
#include "tdsystem.h"  #include "audiowriterinterface.h" ×
#include "synchronizable.h" #include "fft.h"
```

Classes

- class SpectrumAnalyzer

  Spectrum Analyzer.

7.129 streamdescription.cc File Reference

```
#include "tdsystem.h" #include "streamdescription.h"
```

7.130 streamdescription.h File Reference

```
#include "tdsystem.h"  #include "servicelevelagreement.h"
#include "trafficclassvalues.h" #include "managedstreaminterface.-
h" #include "abstractqosdescription.h" #include "sessiondescription.-
h"
```

Classes

- class StreamDescription

  Stream Description.

Variables

- static const cardinal MaxRUEntries = 256

7.130.1 Variable Documentation

7.130.1.1 const cardinal MaxRUEntries = 256 [static]

Maximum number of entries in the list.

7.131 synchronizable.cc File Reference

```
#include "tdsystem.h" #include "synchronizable.h" #include
"thread.h" #include <pthread.h> #include <signal.h>
```

7.132 synchronizable.h File Reference

```
#include "tdsystem.h"  #include <sys/types.h>  #include
<pthread.h> #include <set> #include "synchronizable.icc"
```

Classes

- class Synchronizable

  Synchronizable.

7.133 t1.cc File Reference

```
#include "tdsystem.h" #include "tools.h" #include "rtppacket.-
h" #include "rtcppacket.h" #include "audioencoderinterface.-
h" #include "advancedaudiopacket.h" #include "simpleaudiopacket.-
h" #include "audioquality.h" #include "abstractqosdescription.-
h"   #include <netinet/udp.h>   #include <netinet/ip.h> ×
#include <netinet/ip6.h>
```

Classes

- class SimpleAudioQoSDescription

Functions

- void printQualities (const cardinal maxPacketSize)
- void printQualityLevels ()
- int main (int argc, char ∗∗argv)

7.133.1 Function Documentation

7.133.1.1 int main (int *argc,* char ∗∗ *argv*)

7.133.1.2 void printQualities (const cardinal *maxPacketSize*)

7.133.1.3 void printQualityLevels ( )

7.134 tdin6.h File Reference

```
#include "tdsystem.h" #include <linux/types.h>
```

Classes

- struct in6_flowlabel_req

Defines

- #define IPV6_FL_A_GET 0
- #define IPV6_FL_A_PUT 1
- #define IPV6_FL_A_RENEW 2
- #define IPV6_FL_F_CREATE 1
- #define IPV6_FL_F_EXCL 2
- #define IPV6_FL_S_NONE 0
- #define IPV6_FL_S_EXCL 1
- #define IPV6_FL_S_PROCESS 2
- #define IPV6_FL_S_USER 3
- #define IPV6_FL_S_ANY 255
- #define IPV6_FLOWINFO_FLOWLABEL 0x000fffff
- #define IPV6_FLOWINFO_PRIORITY 0x0ff00000
- #define IPV6_PRIORITY_UNCHARACTERIZED 0x0000
- #define IPV6_PRIORITY_FILLER 0x0100
- #define IPV6_PRIORITY_UNATTENDED 0x0200
- #define IPV6_PRIORITY_RESERVED1 0x0300
- #define IPV6_PRIORITY_BULK 0x0400
- #define IPV6_PRIORITY_RESERVED2 0x0500
- #define IPV6_PRIORITY_INTERACTIVE 0x0600
- #define IPV6_PRIORITY_CONTROL 0x0700
- #define IPV6_PRIORITY_8 0x0800
- #define IPV6_PRIORITY_9 0x0900
- #define IPV6_PRIORITY_10 0x0a00
- #define IPV6_PRIORITY_11 0x0b00
- #define IPV6_PRIORITY_12 0x0c00
- #define IPV6_PRIORITY_13 0x0d00
- #define IPV6_PRIORITY_14 0x0e00
- #define IPV6_PRIORITY_15 0x0f00
- #define IPV6_TLV_PAD0 0
- #define IPV6_TLV_PADN 1
- #define IPV6_TLV_ROUTERALERT 20
- #define IPV6_TLV_JUMBO 194
- #define IPV6_FLOWINFO 11
- #define SCM_SRCRT IPV6_RXSRCRT
- #define IPV6_UNICAST_HOPS 16
- #define IPV6_MULTICAST_IF 17
- #define IPV6_MULTICAST_HOPS 18
- #define IPV6_MULTICAST_LOOP 19
- #define IPV6_ROUTER_ALERT 22
- #define IPV6_MTU_DISCOVER 23
- #define IPV6_MTU 24
- #define IPV6_RECVERR 25
- #define IPV6_PMTUDISC_DONT 0
- #define IPV6_PMTUDISC_WANT 1
- #define IPV6_PMTUDISC_DO 2
- #define IPV6_FLOWLABEL_MGR 32
- #define IPV6_FLOWINFO_SEND 33

7.134.1 Define Documentation

7.134.1.1 #define IPV6_FL_A_GET 0

7.134.1.2 #define IPV6_FL_A_PUT 1

7.134.1.3 #define IPV6_FL_A_RENEW 2

7.134.1.4 #define IPV6_FL_F_CREATE 1

7.134.1.5 #define IPV6_FL_F_EXCL 2

7.134.1.6 #define IPV6_FL_S_ANY 255

7.134.1.7 #define IPV6_FL_S_EXCL 1

7.134.1.8 #define IPV6_FL_S_NONE 0

7.134.1.9 #define IPV6_FL_S_PROCESS 2

7.134.1.10 #define IPV6_FL_S_USER 3

7.134.1.11 #define IPV6_FLOWINFO 11

7.134.1.12 #define IPV6_FLOWINFO_FLOWLABEL 0x000fffff

7.134.1.13 #define IPV6_FLOWINFO_PRIORITY 0x0ff00000

7.134.1.14 #define IPV6_FLOWINFO_SEND 33

7.134.1.15 #define IPV6_FLOWLABEL_MGR 32

7.134.1.16 #define IPV6_MTU 24

7.134.1.17 #define IPV6_MTU_DISCOVER 23

7.134.1.18 #define IPV6_MULTICAST_HOPS 18

7.134.1.19 #define IPV6_MULTICAST_IF 17

7.134.1.20 #define IPV6_MULTICAST_LOOP 19

7.134.1.21 #define IPV6_PMTUDISC_DO 2

7.134.1.22 #define IPV6_PMTUDISC_DONT 0

7.134.1.23 #define IPV6_PMTUDISC_WANT 1

7.134.1.24 #define IPV6_PRIORITY_10 0x0a00

7.134.1.25 #define IPV6_PRIORITY_11 0x0b00

7.134.1.26 #define IPV6_PRIORITY_12 0x0c00

7.134.1.27 #define IPV6_PRIORITY_13 0x0d00

7.134.1.28 #define IPV6_PRIORITY_14 0x0e00

7.134.1.29 #define IPV6_PRIORITY_15 0x0f00

7.134.1.30 #define IPV6_PRIORITY_8 0x0800

7.134.1.31 #define IPV6_PRIORITY_9 0x0900

7.134.1.32 #define IPV6_PRIORITY_BULK 0x0400

7.134.1.33 #define IPV6_PRIORITY_CONTROL 0x0700

7.134.1.34 #define IPV6_PRIORITY_FILLER 0x0100

7.134.1.35 #define IPV6_PRIORITY_INTERACTIVE 0x0600

7.134.1.36 #define IPV6_PRIORITY_RESERVED1 0x0300

7.134.1.37 #define IPV6_PRIORITY_RESERVED2 0x0500

7.134.1.38 #define IPV6_PRIORITY_UNATTENDED 0x0200

7.134.1.39 #define IPV6_PRIORITY_UNCHARACTERIZED 0x0000

7.134.1.40 #define IPV6_RECVERR 25

7.134.1.41 #define IPV6_ROUTER_ALERT 22

7.134.1.42 #define IPV6_TLV_JUMBO 194

7.134.1.43 #define IPV6_TLV_PAD0 0

7.134.1.44 #define IPV6_TLV_PADN 1

7.134.1.45 #define IPV6_TLV_ROUTERALERT 20

7.134.1.46 #define IPV6_UNICAST_HOPS 16

7.134.1.47 #define SCM_SRCRT IPV6_RXSRCRT

7.135 tdmessage.h File Reference

```
#include "tdsystem.h" #include "socketaddress.h" #include
<sys/uio.h> #include <sys/types.h> #include <sys/socket.-
h> #include "tdmessage.icc"
```

Classes

- class SocketMessage< size >

  Socket Message.

Functions

- static size_t CSpace (const size_t payloadLength)
- static size_t CLength (const size_t payloadLength)
- static void * CData (cmsghdr *cmsg)
- static cmsghdr * CFirstHeader (msghdr *header)
- static cmsghdr * CNextHeader (msghdr *header, const cmsghdr *cmsg)
- static const void * CData (const cmsghdr *cmsg)
- static const cmsghdr * CFirstHeader (const msghdr *header)
- static const cmsghdr * CNextHeader (const msghdr *header, const cmsghdr *cmsg)

7.135.1 Function Documentation

7.135.1.1 static void* CData (cmsghdr * *cmsg*) [inline, static]

Wrapper for CMSG_DATA macro.

7.135.1.2 static const void* CData (const cmsghdr * *cmsg*) [inline, static]

Wrapper for CMSG_DATA macro (constant version).

7.135.1.3 static cmsghdr* CFirstHeader (msghdr * *header*) [inline, static]

Wrapper for CMSG_FIRSTHDR macro.

7.135.1.4 static const cmsghdr* CFirstHeader (const msghdr * *header*) [inline, static]

Wrapper for CMSG_FIRSTHDR macro (constant version).

7.135.1.5 static size_t CLength (const size_t *payloadLength*) [inline, static]

Wrapper for CMSG_LEN macro.

7.135.1.6 static cmsghdr* CNextHeader (msghdr * *header,* const cmsghdr * *cmsg*) [inline, static]

Wrapper for CMSG_NXTHDR macro.

7.135.1.7 static const cmsghdr* CNextHeader (const msghdr * *header,* const cmsghdr * *cmsg*) [inline, static]

Wrapper for CMSG_NXTHDR macro (constant version).

7.135.1.8 static size_t CSpace (const size_t *payloadLength*) [inline, static]

Wrapper for CMSG_SPACE macro.

7.136 tdsocket.cc File Reference

```
#include "tdsystem.h" #include "tdsocket.h" #include "randomizer.-
h" #include "tdmessage.h" #include <netdb.h> #include
<netinet/in.h> #include <netinet/tcp.h> #include <sys/uio.-
h> #include <sys/socket.h> #include <net/if.h> #include
<arpa/inet.h>
```

Defines

- #define LINUX_PROC_IPV6_FILE "/proc/net/if_inet6"
- #define IPV6_JOIN_GROUP IPV6_ADD_MEMBERSHIP
- #define IPV6_LEAVE_GROUP IPV6_DROP_MEMBERSHIP

Functions

- SocketAddress ** getAddressArray (const SocketAddress **addressArray, cardinal addresses)
- void setAddressArrayPort (SocketAddress **addressArray, cardinal addresses, const card16 port)
- bool filterInternetAddress (const InternetAddress *newAddress, const cardinal flags)

7.136.1 Define Documentation

7.136.1.1 #define IPV6_JOIN_GROUP IPV6_ADD_MEMBERSHIP

7.136.1.2 #define IPV6_LEAVE_GROUP IPV6_DROP_MEMBERSHIP

7.136.1.3 #define LINUX_PROC_IPV6_FILE "/proc/net/if_inet6"

7.136.2 Function Documentation

7.136.2.1 bool filterInternetAddress (const InternetAddress * *newAddress*, const cardinal *flags*)

7.136.2.2 SocketAddress** getAddressArray (const SocketAddress ** *addressArray*, cardinal *addresses*)

7.136.2.3 void setAddressArrayPort (SocketAddress ** *addressArray*, cardinal *addresses*, const card16 *port*)

7.137 tdsocket.h File Reference

```
#include "tdsystem.h" #include "internetaddress.h" #include
"internetflow.h" #include "ext_socket.h" #include <fcntl.-
h> #include "tdsocket.icc"
```

Classes

- class Socket

  Socket.

Variables

- const cardinal UDPHeaderSize = 8
- const cardinal IPv4HeaderSize = 20
- const cardinal IPv6HeaderSize = 40

7.137.1 Variable Documentation

7.137.1.1 const cardinal IPv4HeaderSize = 20

IPv4 header size.

7.137.1.2 const cardinal IPv6HeaderSize = 40

IPv6 header size.

7.137.1.3 const cardinal UDPHeaderSize = 8

UDP header size.

7.138 tdstrings.cc File Reference

```
#include "tdsystem.h"   #include "tdstrings.h"   #include
<ctype.h>
```

Functions

- std::ostream & operator<< (std::ostream &os, const String &string)
- String operator+ (const String &string1, const String &string2)

7.138.1 Function Documentation

7.138.1.1 String operator+ (const String & *string1,* const String & *string2*)

Implementation of + operator.

7.138.1.2 std::ostream& operator<< (std::ostream & *out,* const String & *string*)

Implementation of << operator.

7.139 tdstrings.h File Reference

```
#include "tdsystem.h" #include "tdstrings.icc"
```

Classes

- class String

  String.

Functions

- std::ostream & operator<< (std::ostream &out, const String &string)
- String operator+ (const String &string1, const String &string2)

7.139.1 Function Documentation

7.139.1.1 String operator+ (const String & *string1,* const String & *string2*)

Implementation of + operator.

7.139.1.2 std::ostream& operator<< (std::ostream & *out,* const String & *string*)

Implementation of << operator.

7.140 tdsystem.h File Reference

`#include <stdio.h> #include <stdlib.h> #include <unistd.h> #include <errno.h> #include <string.h> #include <math.h> #include <iostream> #include <endian.h> #include <stdint.h>`

Defines

- #define OS_Linux 1
- #define OS_FreeBSD 2
- #define OS_Darwin 3
- #define OS_SOLARIS 4

Typedefs

- typedef int8_t sbyte
- typedef uint8_t ubyte
- typedef int8_t int8
- typedef uint8_t card8
- typedef int16_t int16
- typedef uint16_t card16
- typedef int32_t int32
- typedef int32 integer
- typedef uint32_t card32
- typedef int64_t int64
- typedef uint64_t card64
- typedef card32 cardinal

7.140.1 Define Documentation

7.140.1.1 #define OS_Darwin 3

7.140.1.2 #define OS_FreeBSD 2

7.140.1.3 #define OS_Linux 1

7.140.1.4 #define OS_SOLARIS 4

7.140.2 Typedef Documentation

7.140.2.1 typedef uint16_t card16

Datatype for storing a 16-bit cardinal.

7.140.2.2 typedef uint32_t card32

Datatype for storing a 32-bit cardinal.

7.140.2.3 typedef uint64_t card64

Datatype for storing a 64-bit cardinal.

7.140.2.4 typedef uint8_t card8

Datatype for storing a 8-bit cardinal.

7.140.2.5 typedef card32 cardinal

Datatype for storing a default-sized cardinal (32 bits minimum).

7.140.2.6 typedef int16_t int16

Datatype for storing a 16-bit integer.

7.140.2.7 typedef int32_t int32

Datatype for storing a 32-bit intger.

7.140.2.8 typedef int64_t int64

Datatype for storing an 64-bit integer.

7.140.2.9 typedef int8_t int8

Datatype for storing an 8-bit integer.

7.140.2.10 typedef int32 integer

Datatype for storing a default-sized integer (32 bits minimum).

7.140.2.11 typedef int8_t sbyte

Datatype for storing a signed char.

7.140.2.12 typedef uint8_t ubyte

Datatype for storing an unsigned char.

7.141 thread.cc File Reference

```
#include "tdsystem.h" #include "thread.h" #include <sys/time.-
h>
```

7.142 thread.h File Reference

```
#include "tdsystem.h" #include "synchronizable.h" #include
"condition.h" #include <pthread.h> #include <set> #include
"thread.icc"
```

Classes

- class Thread

  Thread.

7.143 timedthread.cc File Reference

```
#include "tdsystem.h"  #include "timedthread.h"  #include
"tools.h" #include <sys/time.h> #include <signal.h>
```

7.144 timedthread.h File Reference

```
#include "tdsystem.h" #include "multitimerthread.h" #include
"timedthread.icc"
```

Classes

- class TimedThread

  Timed Thread.

7.145 tools.cc File Reference

```
#include "tdsystem.h"   #include "tdstrings.h"   #include
"tools.h"  #include <pwd.h>  #include <time.h>  #include
<sys/utsname.h>
```

Defines

- #define PRINT_ALLOCATIONS

Functions

- cardinal calculatePacketsPerSecond (const cardinal payloadBytesPerSecond, const cardinal framesPerSecond, const cardinal maxPacketSize, const cardinal headerLength)
- cardinal calculateBytesPerSecond (const cardinal payloadBytesPerSecond, const cardinal framesPerSecond, const cardinal maxPacketSize, const cardinal headerLength)
- bool scanURL (const String &location, String &protocol, String &host, String &path)
- void printTimeStamp (std::ostream &os)
- bool getUserName (char ∗str, const size_t size, const bool realName, uid_t uid)

7.145.1 Define Documentation

7.145.1.1 #define PRINT_ALLOCATIONS

7.145.2 Function Documentation

7.145.2.1 cardinal calculateBytesPerSecond (const cardinal *payloadBytesPerSecond,* const cardinal *framesPerSecond,* const cardinal *maxPacketSize,* const cardinal *headerLength*)

Calculate frames per second.

Asumption: Every frame has it's own packets.

Parameters

payload-BytesPer-Second	Byte rate of payload data.
framesPer-Second	Frame rate.
maxPacket-Size	Maximum size of a packet.
header-Length	Length of header for each frame.

Returns

Total frames per second.

7.145.2.2 cardinal calculatePacketsPerSecond (const cardinal *payloadBytesPerSecond,* const cardinal *framesPerSecond,* const cardinal *maxPacketSize,* const cardinal *headerLength*)

Calculate packets per second.

Asumption: Every frame has it's own packets.

Parameters

payload-BytesPer-Second	Byte rate of payload data.
framesPer-Second	Frame rate.
maxPacket-Size	Maximum size of a packet.
header-Length	Length of header for each frame.

Returns

Total bytes per second.

7.145.2.3 bool getUserName (char * *str,* const size_t *size,* const bool *realName* = `false`, const uid_t *uid* = `getuid()`)

Get user name for given user ID.

Parameters

str	Buffer to store name to.
size	Size of buffer.
realName	true to get real name (e.g. John Miller); false to get user name (e.g. jmiller).
uid	User ID.

Returns

true for success; false otherwise.

7.145.2.4 void printTimeStamp (std::ostream & *os* = `std::cout`)

Print time stamp (date and time) to given output stream.

Parameters

os	Output stream.

7.145.2.5 bool scanURL (const String & *location*, String & *protocol*, String & *host*, String & *path*)

Scan protocol, host and path from an URL string. The protocol my be missing, if the String "protocol" is initialized with a default.

Parameters

location	String with URL.
protocol	Place to store the protocol name.
host	Place to store the host name.
path	Place to store the path.

Returns

true on success; false otherwise.

7.146 tools.h File Reference

```
#include "tdsystem.h"    #include "tdstrings.h"    #include
"tools.icc"
```

Functions

- void debug (const char *string)

- card64 getMicroTime ()
- card16 translate16 (const card16 x)
- card32 translate32 (const card32 x)
- card64 translate64 (const card64 x)
- card64 translateToBinary (const double x)
- double translateToDouble (const card64 x)
- cardinal calculatePacketsPerSecond (const cardinal payloadBytesPerSecond, const cardinal framesPerSecond, const cardinal maxPacketSize, const cardinal headerLength)
- cardinal calculateBytesPerSecond (const cardinal payloadBytesPerSecond, const cardinal framesPerSecond, const cardinal maxPacketSize, const cardinal headerLength)
- bool scanURL (const String &location, String &protocol, String &host, String &path)
- bool getUserName (char *str, const size_t size, const bool realName=false, const uid_t uid=getuid())
- template<class T >
  void quickSort (T *array, const integer start, const integer end)
- template<class T >
  void quickSortPtr (T *array, const integer start, const integer end, bool(*lt)(T, T), bool(*gt)(T, T))
- template<class T >
  void quickSortGroupPtr (T *array, const integer start, const integer end, bool(*lt)(T, T), bool(*gt)(T, T), bool(*geq)(T, T))
- template<class T >
  cardinal removeDuplicates (T *array, const cardinal length)
- void printTimeStamp (std::ostream &os=std::cout)

7.146.1 Function Documentation

7.146.1.1 cardinal calculateBytesPerSecond (const cardinal *payloadBytesPerSecond,* const cardinal *framesPerSecond,* const cardinal *maxPacketSize,* const cardinal *headerLength*)

Calculate frames per second.

Asumption: Every frame has it's own packets.

Parameters

payload-BytesPer-Second	Byte rate of payload data.
framesPer-Second	Frame rate.
maxPacket-Size	Maximum size of a packet.
header-Length	Length of header for each frame.

Returns

Total frames per second.

7.146.1.2 cardinal calculatePacketsPerSecond (const cardinal *payloadBytesPerSecond,* const cardinal *framesPerSecond,* const cardinal *maxPacketSize,* const cardinal *headerLength*)

Calculate packets per second.

Asumption: Every frame has it's own packets.

Parameters

payload-BytesPer-Second	Byte rate of payload data.
framesPer-Second	Frame rate.
maxPacket-Size	Maximum size of a packet.
header-Length	Length of header for each frame.

Returns

Total bytes per second.

7.146.1.3 void debug (const char * *string*) [inline]

Debug output.

Parameters

string	Debug string to be written to cerr.

7.146.1.4 card64 getMicroTime ( ) [inline]

Get microseconds since January 01, 1970.

Returns

Microseconds since January 01, 1970.

7.146.1.5 bool getUserName (char * *str*, const size_t *size*, const bool *realName* = `false`, const uid_t *uid* = `getuid()`)

Get user name for given user ID.

Parameters

str	Buffer to store name to.
size	Size of buffer.
realName	true to get real name (e.g. John Miller); false to get user name (e.g. jmiller).
uid	User ID.

Returns

true for success; false otherwise.

7.146.1.6 void printTimeStamp (std::ostream & *os* = `std::cout`)

Print time stamp (date and time) to given output stream.

Parameters

os	Output stream.

7.146.1.7 template<class T > void quickSort (T * *array*, const integer *start*, const integer *end*)

Sort array using QuickSort algorithm.

Parameters

array	Array to be sorted.
start	Start offset in array.
end	End offset in array.

7.146.1.8 template<class T > void quickSortGroupPtr (T * *array*, const integer *start*, const integer *end*, bool(*)(T, T) *lt*, bool(*)(T, T) *gt*, bool(*)(T, T) *geq*)

Sort pointer array using QuickSort algorithm.

Parameters

array	Array to be sorted.
start	Start offset in array.
end	End offset in array.

lt	Less than comparision routine for sorting.
gt	Greater than comparision routine for sorting.
geq	Equal routine for separation of groups.

7.146.1.9 template<class T > void quickSortPtr (T * *array,* const integer *start,* const integer *end,* bool(*)(T, T) *lt,* bool(*)(T, T) *gt*)

Sort pointer array using QuickSort algorithm.

Parameters

array	Array to be sorted.
start	Start offset in array.
end	End offset in array.
lt	Less than comparision routine.
gt	Greater than comparision routine.

7.146.1.10 template<class T > cardinal removeDuplicates (T * *array,* const cardinal *length*)

Remove duplicates from *sorted* array.

Parameters

array	Array to be sorted.
length	Length of array.

7.146.1.11 bool scanURL (const String & *location,* String & *protocol,* String & *host,* String & *path*)

Scan protocol, host and path from an URL string. The protocol my be missing, if the String "protocol" is initialized with a default.

Parameters

location	String with URL.
protocol	Place to store the protocol name.
host	Place to store the host name.
path	Place to store the path.

Returns

true on success; false otherwise.

7.146.1.12 card16 translate16 (const card16 *x*) [inline]

Translate 16-bit value to network byte order.

Parameters

x	Value to be translated.

Returns

Translated value.

7.146.1.13 card32 translate32 (const card32 *x*) [inline]

Translate 32-bit value to network byte order.

Parameters

x	Value to be translated.

Returns

Translated value.

7.146.1.14 card64 translate64 (const card64 *x*) [inline]

Translate 64-bit value to network byte order.

Parameters

x	Value to be translated.

Returns

Translated value.

7.146.1.15 card64 translateToBinary (const double *x*) [inline]

Translate double to 64-bit binary.

Parameters

x	Value to be translated.

Returns

Translated value.

7.146.1.16 double translateToDouble (const card64 *x*) [inline]

Translate 64-bit binary to double.

Parameters

x	Value to be translated.

Returns

Translated value.

7.147 trafficclassvalues.cc File Reference

```
#include "tdsystem.h" #include "trafficclassvalues.h"
```

7.148 trafficclassvalues.h File Reference

```
#include "tdsystem.h" #include "trafficclassvalues.icc"
```

Classes

- class TrafficClassValues

  Traffic Class Values.

7.149 trafficshaper.cc File Reference

```
#include "tdsystem.h" #include "trafficshaper.h" #include
"tools.h" #include <algorithm>
```

7.150 trafficshaper.h File Reference

```
#include "tdsystem.h" #include "tdsocket.h" #include "internetaddress.-
h" #include "timedthread.h" #include "trafficclassvalues.-
h" #include <set> #include <deque> #include <vector> ×
#include "trafficshaper.icc"
```

Classes

- class TrafficShaperSingleton
  Traffic Shaper Singleton.
- class TrafficShaper
  Traffic Shaper.
- struct TrafficShaper::TrafficShaperPacket

7.151 unixaddress.cc File Reference

```
#include "tdsystem.h" #include "strings.h" #include "unixaddress.-
h"
```

7.152 unixaddress.h File Reference

```
#include "tdsystem.h" #include "strings.h" #include "socketaddress.-
h" #include <sys/socket.h> #include <sys/un.h> #include
"unixaddress.icc"
```

Classes

- class UnixAddress
  Socket Address.

7.153 wavaudioreader.cc File Reference

```
#include "tdsystem.h" #include "wavaudioreader.h"
```

7.154 wavaudioreader.h File Reference

```
#include "tdsystem.h" #include "audioreaderinterface.h" ×
#include "audioquality.h"
```

Classes

- class WavAudioReader
  WAV Audio Reader.
- struct WavAudioReader::RIFF_Header
- struct WavAudioReader::RIFF_Chunk
- struct WavAudioReader::WAVE_Format