RTP Trace System

1.0

Generated by Doxygen 1.7.6.1

Fri Aug 24 2012 23:51:23

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 Coral Namespace Reference

Namespaces

- namespace Coral
- namespace RTPConstants

Classes

- class AbstractLayerDescription

  Abstract Layer Description.
- class AbstractQoSDescription

  Abstract QoS Description.
- struct BandwidthInfo

  Bandwidth Info.
- struct ResourceUtilizationSimplePoint

  Resource Utilization Simple Point.
- struct ResourceUtilizationMultiPoint

  Resource Utilization Simple Point.
- class BandwidthManager

  Bandwidth Manager.
- class ConstantBitrateFrameSizeScalability

  Constant Bitrate Frame Size Scalability.
- struct DecoderPacket

  DecoderPacket.
- class DecoderInterface

  Decoder Interface.
- class DecoderRepositoryInterface

  Decoder Repository.

- struct EncoderPacket

  EncoderPacket.
- class EncoderInterface

  Encoder Interface.
- class EncoderRepositoryInterface

  Encoder Repository Interface.
- class FrameRateScalabilityInterface

  Frame Rate Scalability Interface.
- class FrameSizeScalabilityInterface

  Frame Rate Scalability Interface.
- class GenericFrameSizeScalability

  Generic Frame Size Scalability.
- class GNUPlotData

  GNUPlot Data.
- class GNUPlotScript

  GNUPlot Data.
- class H263QoSDescription

  H263 QoS Description.
- class H263TraceArray

  H263 Trace Array.
- class H263WriterQoSDescription

  H263 Writer QoS Description.
- class InternetAddress

  Socket Address.
- class InternetFlow

  Internet Flow.
- struct in6_flowlabel_req
- class ManagedStreamInterface

  Managed Stream Interface.
- class MediaInfo

  Media Info.
- class MP3QoSDescription

  MP3 QoS Description.
- class MP3TraceArray

  MP3 Trace Array.
- class MP3WriterQoSDescription

  MP3 Writer QoS Description.
- class MPEGQoSDescription

  MPEG QoS Description.
- class MPEGTraceArray

  MPEG Trace Array.
- class MPEGWriterQoSDescription

  MPEG Writer QoS Description.

- struct PingerHost

  PingerHost.
- class PortableAddress

  Portable Internet Address.
- class Randomizer

  Randomizer.
- struct LayerClassMappingPossibility

  Layer Class Mapping Possibility.
- struct LayerClassMapping

  Layer Class Mapping.
- class ResourceUtilizationPoint

  Resource Utilization Point.
- struct icmp_filter
- class RoundTripTimePinger

  Round Trip Time Pinger.
- class RTCPAbstractServer

  RTCP abstract server.
- class RTCPCommonHeader

  RTCP Common Header.
- class RTCPSenderInfoBlock

  RTCP Sender Info Block.
- class RTCPReceptionReportBlock

  RTCP Reception Report Block.
- class RTCPReport

  RTCP Report.
- class RTCPSenderReport

  RTCP Sender Report.
- class RTCPReceiverReport

  RTCP Sender Report.
- class RTCPSourceDescriptionItem

  RTCP Source Description Item.
- class RTCPSourceDescriptionChunk

  RTCP Source Description Chunk.
- class RTCPSourceDescription

  RTCP Source Description (SDES)
- class RTCPBye

  RTCP BYE Message.
- class RTCPApp

  RTCP APP Message.
- class RTCPReceiver

  RTCP Receiver.
- class RTCPSender

  RTCP Sender.

- class RTPPacket

  RTP Packet.
- class RTPReceiver

  RTP Receiver.
- class RTPSender

  RTP Sender.
- class SeqNumValidator

  Sequence Number Validator.
- struct DiffServClass

  DiffServ Class.
- class ServiceLevelAgreement

  Trace Layer Configuration.
- struct SessionDescription

  Session Description.
- class Socket

  Socket.
- class SocketAddress

  Socket Address.
- class SourceStateInfo

  Source State Info.
- class StreamDescription

  Stream Description.
- class Synchronizable

  Synchronizable.
- struct TDTFPrefixExtensionHeader

  TDTF Prefix Extension Header.
- struct TDTFPrefix

  TDTF Prefix.
- struct TDTFSuffix

  TDTF Suffix.
- struct EmpiricalEnvelopePair

  Empirical Envelope Pair.
- struct EmpiricalEnvelope

  Empirical Envelope Header.
- struct FrameDescription

  Frame Description.
- struct TraceHeader

  Trace Header.
- struct UtilizationHeader

  Trace Header.
- struct ResourceUtilizationEntry

  Resource Utilization Entry.
- struct ResourceUtilizationHeader

Resource Utilization Header.

- struct IntervalHeader

  Interval Header.
- struct LayerHeader

  Layer Header.
- struct PositionLengthIntervalIndexEntry

  Layer Header.
- struct PositionLengthIntervalIndexHeader

  Layer Header.
- struct ResourceUtilizationListIndexEntry

  Resource Utilization List Index Entry.
- struct ResourceUtilizationListIndexHeader

  Resource Utilization List Index Header.
- struct MainIndexEntry

  Layer Header.
- struct MainIndexHeader

  Main Index Header.
- class TDTFMediaReader

  TDTF Media Reader.
- class TDTFReader

  Trace Reader.
- class TDTFWriter

  TDTF Writer.
- class Thread

  Thread.
- class TimedThread

  Timed Thread.
- class TraceArray

  Trace Array.
- class TraceClient

  Trace Client.
- class TraceClientAppPacket

  Trace Client RTCP-SDES-APP-PRIV Packet.
- struct TraceLayerConfiguration

  Trace Layer Configuration.
- struct TraceConfiguration

  Trace Configuration.
- class TraceDecoder

  Trace Decoder.
- class TraceDecoderInterface

  Trace Decoder Interface.
- class TraceDecoderRepository

  Trace Decoder Repository.

- struct QualityScenarioEntry
- struct QualityScenario
- class TraceEncoder

  Trace Encoder.
- class TraceEncoderInterface

  Trace Encoder Interface.
- class TraceEncoderRepository

  Trace Encoder Repository.
- class TraceFrameRateScalability

  Trace Frame Rate Scalability.
- class TraceFrameSizeScalability

  Trace Frame Size Scalability.
- class TracePacket

  Trace Packet.
- struct TracePacketData

  Trace Packet Data.
- class TraceLayerDescription

  Trace Layer QoS Description.
- class TraceQoSDescription

  Trace QoS Description.
- class TraceServer

  Trace Server.
- class TrafficClassValues

  Traffic Class Values.
- class TrafficPolicer

  Traffic Policer.
- class TrafficShaperSingleton

  Traffic Shaper Singleton.
- class TrafficShaper

  Traffic Shaper.
- class UnixAddress

  Socket Address.

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 }
- 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 }

- enum UtilityFunctions { UF_Linear = 0x0000, UF_Exponential1 = 0x0010, UF_-Exponential2 = 0x0011, UF_Undefined = 0xffff }

Functions

- ostream & operator<< (ostream &os, const AbstractQoSDescription &aqd)
- ostream & operator<< (ostream &os, const BandwidthInfo &bi)
- ostream & operator<< (ostream &os, const ResourceUtilizationSimplePoint &srup)
- ostream & operator<< (ostream &os, const ResourceUtilizationMultiPoint &srup)
- void breakDetector (int signum)
- void installBreakDetector ()
- void uninstallBreakDetector ()
- bool breakDetected ()
- ostream & operator<< (ostream &os, const MediaInfo &mi)
- int operator== (const PingerHost &ph1, const PingerHost &ph2)
- int operator< (const PingerHost &ph1, const PingerHost &ph2)
- int operator> (const PingerHost &ph1, const PingerHost &ph2)
- ostream & operator<< (ostream &os, const ResourceUtilizationPoint &rup)
- ostream & operator<< (ostream &os, RoundTripTimePinger &pinger)
- ostream & operator<< (ostream &os, const RTPPacket &packet)
- ostream & operator<< (ostream &os, const ServiceLevelAgreement sla)
- ostream & operator<< (ostream &os, const SocketAddress &sa)
- card64 getMicroTime ()
- 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 (ostream &os)
- void debug (const char *string)
- 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)
- template<class T >
  void quickSort (T *array, const integer start, const integer end)
- template<class T >
  cardinal removeDuplicates (T *array, const cardinal length)
- ostream & operator<< (ostream &os, const TraceArray &traceArray)
- ostream & operator<< (ostream &os, const TraceConfiguration &config)
- double evaluateUtilityFunction (const cardinal type, const double scaleFactor, const double *constantArray, const cardinal constants)
- double evaluateUtilityFunctionTranslated (const cardinal type, const double scale-Factor, const card64 *constantArray, const cardinal constants)

Variables

- bool DetectedBreak = false
- bool PrintedBreak = false
- TraceConfiguration TraceConfig
- const card64 PositionStepsPerSecond = (card64)1000000000
- const cardinal UDPHeaderSize = 8
- const cardinal IPv4HeaderSize = 20
- const cardinal IPv6HeaderSize = 40
- const card8 TraceServerDefaultTrafficClass = 0x00
- const card8 TraceClientDefaultTrafficClass = 0x00
- QualityScenario QualityScenarios []

5.1.1 Enumeration Type Documentation

5.1.1.1 enum Coral::MediaError

Definition of encoder errors.

Enumerator:

ME_NoError

ME_NoMedia

ME_EOF

ME_UnrecoverableError

ME_BadMedia

ME_ReadError

ME_OutOfMemory

5.1.1.2 enum Coral::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

5.1.1.3 enum Coral::RTCP_Type

Definition of RTCP message types.

Enumerator:

RTCP_SR

RTCP_RR

RTCP_SDES

RTCP_BYE

RTCP_APP

5.1.1.4 enum Coral::UtilityFunctions

This is an enumeration of predefined utility function types.

Enumerator:

UF_Linear Linear: U(x) = x. Parameters: none

UF_Exponential1 Exponential #1 (from [LS98]): U(x) = 1 - exp(a∗x + b). a = (ln(-0.95 + 1) - b) / q95 b = (q95 ∗ ln(-0.05 + 1) - q50 ∗ ln(-0.95 + 1)) / (q95 - q50)

Parameters: q50 = Scale factor for 50% perceptual quality. q95 = Scale facotr for 95% perceptual quality.

UF_Exponential2 Exponential #2 (from [Rog98]): U(x) = a∗ln(b∗x + c). U(1) = 0, U(2) = 1. This is equal to U(x) = a∗ln(b∗(x+1) + c) with U(0) = 0, U(1) = 1. a = 1 / (p - 10) b = exp(1/a) - 1 c = 2 - exp(1/a)

Parameters: p = Sensitivity

UF_Undefined Undefined.

5.1.2 Function Documentation

5.1.2.1 bool Coral::breakDetected ( )

Check, if break has been detected.

5.1.2.2 void Coral::breakDetector (int *signum*)

5.1.2.3 cardinal Coral::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.

5.1.2.4 cardinal Coral::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.

5.1.2.5 void Coral::debug (const char * *string*) [inline]

Debug output.

Parameters

string	Debug string to be written to cerr.

5.1.2.6 double Coral::evaluateUtilityFunction (const cardinal *type*, const double *scaleFactor*, const double * *constantArray*, const cardinal *constants*)

Evaluate utility function of given type.

Parameters

type	Utility function type.
scaleFactor	Scale factor to evaluate utility function for (out of [0,1]).
constant-Array	Array of utility function's constants.
constants	Number of constants.

Returns

Result.

5.1.2.7 double Coral::evaluateUtilityFunctionTranslated (const cardinal *type*, const double *scaleFactor*, const card64 * *constantArray*, const cardinal *constants*) [inline]

Evaluate utility function of given type using byte-order translated constants (translate-ToDouble()).

Parameters

type	Utility function type.
scaleFactor	Scale factor to evaluate utility function for (out of [0,1]).
constant-Array	Array of utility function's constants.
constants	Number of constants.

Returns

Result.

See also

translateToDouble
translateToBinary

5.1.2.8 card64 Coral::getMicroTime ( )

Get microseconds since January 01, 1970.

Returns

Microseconds since January 01, 1970.

5.1.2.9 void Coral::installBreakDetector ( )

Install break handler.

5.1.2.10 int Coral::operator< (const PingerHost & *ph1,* const PingerHost & *ph2*) [inline]

Operator "<".

5.1.2.11 ostream& Coral::operator<< (ostream & *os,* const TraceConfiguration & *config*)

Output operator.

5.1.2.12 ostream& Coral::operator<< (ostream & *os,* const ServiceLevelAgreement *config*)

Output operator.

5.1.2.13 ostream& Coral::operator<< (ostream & *os,* const RTPPacket & *packet*)

Output operator.

5.1.2.14 ostream& Coral::operator<< (ostream & *os,* const BandwidthInfo & *bi*)

Operator "<<".

5.1.2.15 ostream& Coral::operator<< (ostream & *os,* const ResourceUtilizationSimplePoint & *srup*)

Output operator.

5.1.2.16 ostream& Coral::operator<< (ostream & *os,* const ResourceUtilizationPoint & *rup*)

Output operator.

5.1.2.17 ostream& Coral::operator<< (ostream & *os,* const MediaInfo & *mi*)

Output operator.

5.1.2.18 ostream& Coral::operator<< (ostream & *os,* const ResourceUtilizationMultiPoint & *srup*)

Output operator.

5.1.2.19 ostream& Coral::operator<< (ostream & *os,* const SocketAddress & *sa*) [inline]

Output operator.

5.1.2.20 ostream& Coral::operator<< (ostream & *os,* const TraceArray & *traceArray*)

Output operator.

5.1.2.21 ostream& Coral::operator<< (ostream & *os,* const AbstractQoSDescription & *aqd*)

Output operator.

5.1.2.22 ostream& Coral::operator<< (ostream & *os,* RoundTripTimePinger & *pinger*)

Friend output operator.

5.1.2.23 int Coral::operator== (const PingerHost & *ph1,* const PingerHost & *ph2*) [inline]

Operator "==".

Operator "!=".

5.1.2.24 int Coral::operator> (const PingerHost & *ph1,* const PingerHost & *ph2*) [inline]

Operator ">".

5.1.2.25 void Coral::printTimeStamp (ostream & *os* = cout)

Print time stamp (date and time) to given output stream.

Parameters

os	Output stream.

5.1.2.26 template<class T > void Coral::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.

5.1.2.27 template<class T > cardinal Coral::removeDuplicates (T * *array*, const cardinal *length*)

Remove duplicates from *sorted* array.

Parameters

array	Array to be sorted.
length	Length of array.

5.1.2.28 bool Coral::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.

5.1.2.29 card16 Coral::translate16 (const card16 *x*) [inline]

Translate 16-bit value to network byte order.

Parameters

x	Value to be translated.

Returns

Translated value.

5.1.2.30 card32 Coral::translate32 (const card32 *x*) [inline]

Translate 32-bit value to network byte order.

Parameters

x	Value to be translated.

Returns

Translated value.

5.1.2.31 card64 Coral::translate64 (const card64 *x*) [inline]

Translate 64-bit value to network byte order.

Parameters

x	Value to be translated.

Returns

Translated value.

5.1.2.32 card64 Coral::translateToBinary (const double *x*) [inline]

Translate double to 64-bit binary.

Parameters

x	Value to be translated.

Returns

Translated value.

5.1.2.33 double Coral::translateToDouble (const card64 *x*) [inline]

Translate 64-bit binary to double.

Parameters

x	Value to be translated.

Returns

Translated value.

5.1.2.34 void Coral::uninstallBreakDetector ( )

Uninstall break handler.

5.1.3 Variable Documentation

5.1.3.1 bool Coral::DetectedBreak = false

5.1.3.2 const cardinal Coral::IPv4HeaderSize = 20

IPv4 header size.

5.1.3.3 const cardinal Coral::IPv6HeaderSize = 40

IPv6 header size.

5.1.3.4 const card64 Coral::PositionStepsPerSecond = (card64)1000000000

Constant for position steps per second: 1 step = 1 nanosecond;

5.1.3.5 bool Coral::PrintedBreak = false

5.1.3.6 QualityScenario Coral::QualityScenarios[]

5.1.3.7 const card8 Coral::TraceClientDefaultTrafficClass = 0x00

Default traffic class/TOS for RTCP control connection from client to server.

5.1.3.8 TraceConfiguration Coral::TraceConfig

Global trace configuration, filled with default values.

5.1.3.9 const card8 Coral::TraceServerDefaultTrafficClass = 0x00

Default traffic class/TOS for RTP data connection from server to client.

5.1.3.10 const cardinal Coral::UDPHeaderSize = 8

UDP header size.

5.2 Coral::Coral Namespace Reference

5.3 Coral::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

5.3.1 Variable Documentation

5.3.1.1 const cardinal Coral::RTPConstants::RTPDefaultHeaderSize = 12

Default RTP header size (CC = 0).

5.3.1.2 const cardinal Coral::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.

5.3.1.3 const cardinal Coral::RTPConstants::RTPMaxPayloadLimit = 8192

RTP maximum payload limit.

5.3.1.4 const cardinal Coral::RTPConstants::RTPMaxQualityLayers = 16

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

5.3.1.5 const double Coral::RTPConstants::RTPMicroSecondsPerTimeStamp = 1000.0 / 16.0

Constant for microseconds per RTP timestamp.

5.3.1.6 const card8 Coral::RTPConstants::RTPVersion = 2

Constant for RTP version.

Chapter 6

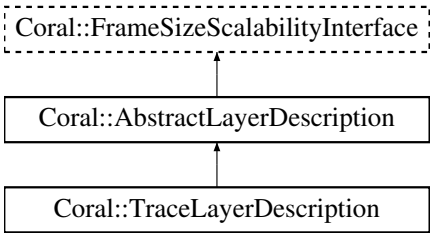
Class Documentation

6.1 Coral::AbstractLayerDescription Class Reference

Abstract Layer Description.

`#include <abstractlayerdescription.h>`

Inheritance diagram for Coral::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 Coral::AbstractLayerDescription::LayerFlags

Layer flags.

Enumerator:

LF_BaseLayer

LF_ExtensionLayer

6.1.3 Constructor & Destructor Documentation

6.1.3.1 Coral::AbstractLayerDescription::AbstractLayerDescription ( )

Constructor.

6.1.3.2 Coral::AbstractLayerDescription::~AbstractLayerDescription ( ) [virtual]

Destructor.

6.1.4 Member Function Documentation

6.1.4.1 card64 Coral::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 Coral::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 Coral::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 Coral::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 Coral::AbstractLayerDescription::getBandwidth ( ) const [inline]

Get bandwidth.

Returns

Bandwidth.

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

Get buffer delay.

Returns

Buffer delay in frame rate units.

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

Get destination address.

Returns

Destination address.

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

Get flags.

6.1.4.9 double Coral::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 Coral::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 Coral::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 Coral::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 Coral::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 Coral::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 Coral::AbstractLayerDescription::getMaxJitter ( ) const [inline]

Get maximum jitter.

Returns

Maximum jitter in microseconds.

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

Get maximum loss rate.

Returns

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

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

Get maximum transfer delay.

Returns

Maximum transfer delay in microseconds.

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

Get packet rate.

Parameters

frameRate	Frame rate.

Returns

Bandwidth limit.

6.1.4.27 cardinal Coral::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 Coral::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 Coral::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 Coral::AbstractLayerDescription::getPrevFrameSizeForDelay-AndSize (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 Coral::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 Coral::AbstractLayerDescription::getSource ( ) const [inline]

Get source address.

Returns

Source address.

6.1.4.33 void Coral::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

pktHeaderSize	Packet header size, e.g. 40 + 8 + 12 (IPv6 + UDP + RTP).
pktMaxSize	Maximum packet size, e.g. 1500.
maxTransferDelay	Maximum transfer delay in microseconds.
maxBufferDelay	Maximum buffer delay in frame rate units.
maxLossRate	Maximum loss rate (out of [0,1]).
maxJitter	Maximum jitter in microseconds.

6.1.4.34 bool Coral::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 Coral::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.
newBufferDelay	Output buffer delay.

Returns

Output payload bandwidth.

6.1.4.36 cardinal Coral::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 Coral::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 Coral::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 Coral::AbstractLayerDescription::setBufferDelay (const cardinal *bufferDelay*) [inline]

Set buffer delay.

Parameters

bufferDelay	Buffer delay in frame rate units.

Returns

Buffer delay set in frame rate units.

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

Set destination address.

Parameters

destination	Destination address

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

Set flags.

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

Get maximum jitter.

Parameters

maxJitter	Maximum jitter in microseconds.

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

Set maximum transfer delay.

Parameters

maxDelay	Maximum transfer delay in microseconds.

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

Set source address.

Parameters

source	Source address.

6.1.5 Member Data Documentation

6.1.5.1 card64 Coral::AbstractLayerDescription::Bandwidth [protected]

6.1.5.2 cardinal Coral::AbstractLayerDescription::BufferDelay [protected]

6.1.5.3 InternetFlow Coral::AbstractLayerDescription::Destination [protected]

6.1.5.4 cardinal Coral::AbstractLayerDescription::Flags [protected]

6.1.5.5 cardinal Coral::AbstractLayerDescription::MaxBufferDelay [protected]

6.1.5.6 double Coral::AbstractLayerDescription::MaxJitter [protected]

6.1.5.7 double Coral::AbstractLayerDescription::MaxLossRate [protected]

6.1.5.8 double Coral::AbstractLayerDescription::MaxTransferDelay [protected]

6.1.5.9 cardinal Coral::AbstractLayerDescription::PktHeaderSize [protected]

6.1.5.10 cardinal Coral::AbstractLayerDescription::PktMaxSize [protected]

6.1.5.11 InternetAddress Coral::AbstractLayerDescription::Source [protected]

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

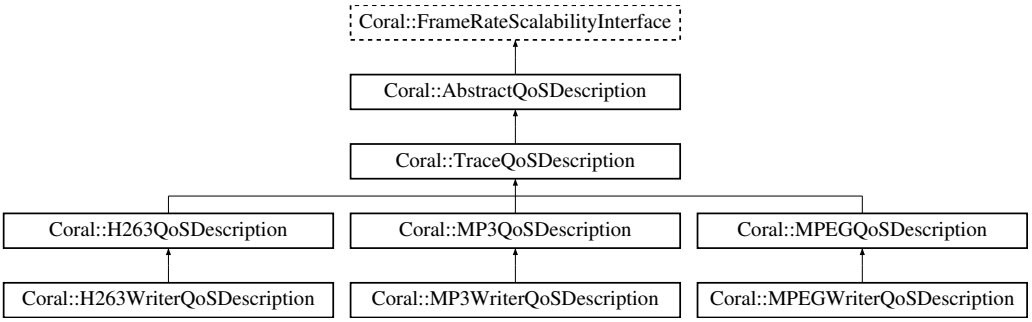
- abstractlayerdescription.h
- abstractlayerdescription.cc

6.2 Coral::AbstractQoSDescription Class Reference

Abstract QoS Description.

`#include <abstractqosdescription.h>`

Inheritance diagram for Coral::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

6.2.1 Detailed Description

Abstract QoS Description.

This class contains a stream's QoS requirements.

Author

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

6.2.2 Constructor & Destructor Documentation

6.2.2.1 Coral::AbstractQoSDescription::AbstractQoSDescription ( )

Constructor.

6.2.2.2 Coral::AbstractQoSDescription::~AbstractQoSDescription ( ) [virtual]

Destructor.

6.2.3 Member Function Documentation

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

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

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

Parameters

totalBandwidth	Total bandwidth.
rup	ResourceUtilizationPoint reference to store result.

Returns

Utilization.

See also

calculateMaxUtilizationForBandwidthArray

6.2.3.3 void Coral::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.

Reimplemented in Coral::H263QoSDescription, Coral::MP3QoSDescription, and Coral::MPEGQoSDescription.

6.2.3.4 cardinal Coral::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 getPrecomputedResourceUtilizationList().

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

6.2.3.5 double Coral::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.

Reimplemented in Coral::TraceQoSDescription.

6.2.3.6 void Coral::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]

6.2.3.7 double Coral::AbstractQoSDescription::getFrameRate ( ) const [inline]

Get frame rate.

Returns

Frame rate.

6.2.3.8 double Coral::AbstractQoSDescription::getFrameRateScaleFactor ( ) const [inline]

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

Returns

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

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

Get layer.

Parameters

layer	Layer number.

Returns

Layer.

Implemented in Coral::TraceQoSDescription.

6.2.3.10 virtual cardinal Coral::AbstractQoSDescription::getLayers ( ) const [pure virtual]

Get number of layers.

Returns

Number of layers.

Implemented in Coral::TraceQoSDescription.

6.2.3.11 card64 Coral::AbstractQoSDescription::getMaxBandwidth ( ) const

Get maximum required total bandwidth.

Returns

Maximum total bandwidth.

6.2.3.12 card64 Coral::AbstractQoSDescription::getMaxWantedBandwidth ( ) const

Get maximum wanted bandwidth.

Returns

Maximum wanted bandwidth.

6.2.3.13 card64 Coral::AbstractQoSDescription::getMinBandwidth ( ) const

Get minimum required total bandwidth.

Returns

Minimum total bandwidth.

6.2.3.14 card64 Coral::AbstractQoSDescription::getMinWantedBandwidth ( ) const

Get minimum wanted bandwidth.

Returns

Minimum wanted bandwidth.

6.2.3.15 double Coral::AbstractQoSDescription::getNextFrameRate ( ) const [inline]

Get next higher frame rate.

Returns

Frame rate.

6.2.3.16 card64 Coral::AbstractQoSDescription::getPosition ( ) const [inline]

Get position.

Returns

Position.

6.2.3.17 virtual cardinal Coral::AbstractQoSDescription::getPrecomputed-ResourceUtilizationList (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

Implemented in Coral::TraceQoSDescription.

6.2.3.18 double Coral::AbstractQoSDescription::getPrevFrameRate ( ) const [inline]

Get next lower frame rate.

Returns

Frame rate.

6.2.3.19 double Coral::AbstractQoSDescription::getResources (ResourceUtilizationPoint & *rup*) const

Get resources.

Parameters

rup	ResourceUtilizationPoint reference to store resources.

Returns

Utilization.

6.2.3.20 int8 Coral::AbstractQoSDescription::getSessionPriority ( ) const [inline]

Get session priority.

Returns

Session priority.

6.2.3.21 int8 Coral::AbstractQoSDescription::getStreamPriority ( ) const [inline]

Get stream priority.

Returns

Stream priority.

6.2.3.22 double Coral::AbstractQoSDescription::getWantedUtilization ( ) const [inline]

Get wanted utilization.

Returns

Wanted utilization.

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

Initialize description.

Parameters

frameRate	Frame rate.

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

Set frame rate.

Parameters

frameRate	Frame rate.

Returns

Frame rate set.

6.2.3.25 void Coral::AbstractQoSDescription::setMaxWantedBandwidth (const card64 *bandwidth*)

Set maximum wanted bandwidth.

Parameters

wanted	bandwidth Maximum wanted bandwidth.

6.2.3.26 void Coral::AbstractQoSDescription::setMinWantedBandwidth (const card64 *bandwidth*)

Set minimum wanted bandwidth.

Parameters

wanted	bandwidth Minimum wanted bandwidth.

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

Set position.

Parameters

position	Position.

6.2.3.28 double Coral::AbstractQoSDescription::setResources (const ResourceUtilizationPoint & *rup*)

Set resources.

Parameters

rup	ResourceUtilizationPoint reference containing resources.

Returns

Utilization.

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

Set session priority.

Parameters

priority	Session priority.

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

Set stream priority.

Parameters

priority	Stream priority.

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

Set wanted utilization.

Parameters

utilization	Wanted utilization.

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

Update description.

Parameters

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

Implemented in Coral::TraceQoSDescription.

6.2.4 Member Data Documentation

6.2.4.1 double Coral::AbstractQoSDescription::FrameRate [protected]

Reimplemented in Coral::TraceQoSDescription.

6.2.4.2 card64 Coral::AbstractQoSDescription::MaxWantedBandwidth [protected]

6.2.4.3 card64 Coral::AbstractQoSDescription::MinWantedBandwidth [protected]

6.2.4.4 cardinal Coral::AbstractQoSDescription::PktHeaderSize [protected]

6.2.4.5 cardinal Coral::AbstractQoSDescription::PktMaxSize [protected]

6.2.4.6 card64 Coral::AbstractQoSDescription::Position [protected]

Reimplemented in Coral::TraceQoSDescription.

6.2.4.7 int8 Coral::AbstractQoSDescription::SessionPriority [protected]

6.2.4.8 int8 Coral::AbstractQoSDescription::StreamPriority [protected]

6.2.4.9 double Coral::AbstractQoSDescription::WantedUtilization [protected]

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

- abstractqosdescription.h
- abstractqosdescription.cc

6.3 Action Struct Reference

Public Attributes

- Action * NextAction
- card64 Offset
- cardinal ClassID
- cardinal Type
- card64 Bandwidth
- double TransferDelay
- double LossRate
- double Jitter
- card8 TrafficClass

6.3.1 Member Data Documentation

6.3.1.1 card64 Action::Bandwidth

6.3.1.2 cardinal Action::ClassID

6.3.1.3 double Action::Jitter

6.3.1.4 double Action::LossRate

6.3.1.5 Action* Action::NextAction

6.3.1.6 card64 Action::Offset

6.3.1.7 card8 Action::TrafficClass

6.3.1.8 double Action::TransferDelay

6.3.1.9 cardinal Action::Type

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

- tsimulator.cc

6.4 Coral::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

6.4.1 Detailed Description

Bandwidth Info.

This is a description of bandwidth requirements.

Author

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

6.4.2 Member Function Documentation

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

Operator "!=".

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

Operator "==".

6.4.2.3 void Coral::BandwidthInfo::reset ( )

Reset.

6.4.3 Member Data Documentation

6.4.3.1 cardinal Coral::BandwidthInfo::BufferDelay

Buffer delay in microseconds.

6.4.3.2 card64 Coral::BandwidthInfo::BytesPerSecond

Bytes per second.

6.4.3.3 double Coral::BandwidthInfo::MaxJitter

Maximum jitter.

6.4.3.4 double Coral::BandwidthInfo::MaxLossRate

Maximum loss rate.

6.4.3.5 double Coral::BandwidthInfo::MaxTransferDelay

Maximum transfer delay.

6.4.3.6 cardinal Coral::BandwidthInfo::PacketsPerSecond

Packets per second.

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

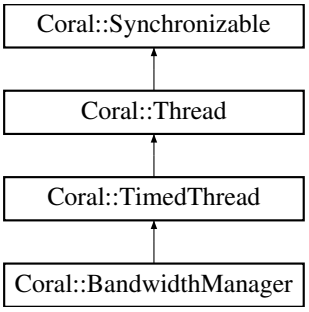
- bandwidthinfo.h
- bandwidthinfo.cc

6.5 Coral::BandwidthManager Class Reference

Bandwidth Manager.

`#include <bandwidthmanager.h>`

Inheritance diagram for Coral::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 RTCPReceptionReportBlock ∗report, const cardinal layer)
- void bufferFlushEvent (ManagedStreamInterface ∗stream, const cardinal layer)
- void timerEvent ()
- void setLogStream (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
- multimap < ManagedStreamInterface ∗, StreamDescription ∗ > StreamSet
- 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
- ostream * Log
- card64 LogStartupTimeStamp
- bool Changed

6.5.1 Detailed Description

Bandwidth Manager.

This is the bandwidth manager.

Author

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

6.5.2 Constructor & Destructor Documentation

6.5.2.1 Coral::BandwidthManager::BandwidthManager (ServiceLevelAgreement * *sla*, RoundTripTimePinger * *rttp*)

Constructor.

Parameters

sla	ServiceLevelAgreement object.
rttp	RoundTripTimePinger object.

6.5.2.2 Coral::BandwidthManager::~BandwidthManager ( )

Destructor.

6.5.3 Member Function Documentation

6.5.3.1 void Coral::BandwidthManager::addStream (ManagedStreamInterface * *stream*, const cardinal *sessionID* = 0, const char * *name* = NULL)

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).

6.5.3.2 void Coral::BandwidthManager::bufferFlushEvent (ManagedStreamInterface * *stream,* const cardinal *layer*)

Buffer flush for a given layer.

Parameters

stream	Stream.

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

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

6.5.3.5 void Coral::BandwidthManager::doCompleteRemapping ( ) [private]

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

6.5.3.7 void Coral::BandwidthManager::forceCompleteRemapping ( ) [inline]

Force a complete remapping.

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

Get fairnes settings.

Parameters

fairness-Session	Reference to store session fairness.
fairness-Stream	Reference to store stream fairness.

6.5.3.9 void Coral::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.

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

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

Get QoS optimization parameters.

Parameters

maxRUPoints	Reference to store maximum number of RU points per stream.
utilizationThreshold	Reference to store utilization threshold.
bandwidthThreshold	Reference to store bandwidth threshold.
systemDelayTolerance	Reference to store the system's delay tolerance for the buffering optimization.

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

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

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

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

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

6.5.3.17 void Coral::BandwidthManager::intervalChangeEvent (ManagedStreamInterface * *stream,* const bool *isNew,* const card64 *when,* const bool *newRUList*)

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.

6.5.3.18 void Coral::BandwidthManager::removeStream (ManagedStreamInterface * *stream*)

Remove stream from management.

Parameters

stream	Stream to remove.

6.5.3.19 void Coral::BandwidthManager::reportEvent (ManagedStreamInterface * *stream,* const RTCPReceptionReportBlock * *report,* const cardinal *layer*)

Report reception for given layer.

Parameters

stream	Stream.
report	Report.
layer	Layer.

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

Set fairnes settings.

Parameters

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

6.5.3.21 void Coral::BandwidthManager::setLogStream (ostream * *logStream*)

Set log stream.

Parameters

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

6.5.3.22 void Coral::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.

6.5.3.23 void Coral::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.

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

6.5.3.25 void Coral::BandwidthManager::timerEvent ( ) [virtual]

Implementation of TimedThread's timerEvent() method.

See also

TimedThread::timerEvent

Implements Coral::TimedThread.

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

6.5.3.27 void Coral::BandwidthManager::updateReservation (StreamDescription * *streamDescription*)

6.5.3.28 void Coral::BandwidthManager::updateStream (ManagedStreamInterface * *stream*)

Update stream.

Parameters

stream	Stream to be updated.

6.5.4 Member Data Documentation

6.5.4.1 double Coral::BandwidthManager::AlphaJitter

6.5.4.2 double Coral::BandwidthManager::AlphaLossRate

6.5.4.3 card64 Coral::BandwidthManager::BandwidthThreshold

6.5.4.4 bool Coral::BandwidthManager::Changed [private]

6.5.4.5 card64 Coral::BandwidthManager::ClassAvailableBandwidthArray[TrafficClassValues::MaxValues]

6.5.4.6 card64 Coral::BandwidthManager::ClassBandwidthArray[TrafficClassValues::MaxValues]

6.5.4.7 cardinal Coral::BandwidthManager::CompleteRemappings

6.5.4.8 bool Coral::BandwidthManager::EnablePartialRemappings

6.5.4.9 double Coral::BandwidthManager::FairnessSession

6.5.4.10 double Coral::BandwidthManager::FairnessStream

6.5.4.11 card64 Coral::BandwidthManager::LastCompleteRemapping

6.5.4.12 card64 Coral::BandwidthManager::LastCompleteRemappingDuration

6.5.4.13 ostream* Coral::BandwidthManager::Log [private]

6.5.4.14 card64 Coral::BandwidthManager::LogStartupTimeStamp [private]

6.5.4.15 card64 Coral::BandwidthManager::MaxRemappingInterval

6.5.4.16 cardinal Coral::BandwidthManager::MaxRUPoints

6.5.4.17 double Coral::BandwidthManager::PartialRemappingPortion

6.5.4.18 cardinal Coral::BandwidthManager::PartialRemappings

6.5.4.19 double Coral::BandwidthManager::PartialRemappingUtilizationTolerance

6.5.4.20 RoundTripTimePinger* Coral::BandwidthManager::RTTP [private]

6.5.4.21 cardinal Coral::BandwidthManager::Sessions

6.5.4.22 multimap<cardinal,SessionDescription*> Coral::BandwidthManager::SessionSet

6.5.4.23 card64 Coral::BandwidthManager::SimulatorTime = 0 [static]

6.5.4.24 ServiceLevelAgreement* Coral::BandwidthManager::SLA

6.5.4.25 int64 Coral::BandwidthManager::SLAUpdateRecommendation[TrafficClassValues::MaxValues]

6.5.4.26 card64 Coral::BandwidthManager::StreamIDGenerator

6.5.4.27 cardinal Coral::BandwidthManager::Streams

6.5.4.28 multimap<ManagedStreamInterface*,StreamDescription*> Coral::BandwidthManager::StreamSet

6.5.4.29 double Coral::BandwidthManager::SystemDelayTolerance

6.5.4.30 card64 Coral::BandwidthManager::TotalAvailableBandwidth

6.5.4.31 card64 Coral::BandwidthManager::TotalBandwidth

6.5.4.32 cardinal Coral::BandwidthManager::TotalBufferFlushes

6.5.4.33 bool Coral::BandwidthManager::UnlayeredAllocation

6.5.4.34 double Coral::BandwidthManager::UtilizationThreshold

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

- bandwidthmanager.h
- bandwidthmanager.cc

6.6 ClassEntry Struct Reference

Public Attributes

- card64 Bandwidth
- double CostFactor
- double Variability
- card8 TrafficClass

6.6.1 Member Data Documentation

6.6.1.1 card64 ClassEntry::Bandwidth

6.6.1.2 double ClassEntry::CostFactor

6.6.1.3 card8 ClassEntry::TrafficClass

6.6.1.4 double ClassEntry::Variability

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

- loganalyzer.cc

6.7 Coral::RTCPAbstractServer::Client Struct Reference

```
#include <rtcpabstractserver.h>
```

Public Attributes

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

6.7.1 Detailed Description

Client structure with information on the client.

6.7.2 Member Data Documentation

6.7.2.1 InternetFlow Coral::RTCPAbstractServer::Client::ClientAddress

6.7.2.2 card32 Coral::RTCPAbstractServer::Client::SSRC

6.7.2.3 card64 Coral::RTCPAbstractServer::Client::Timeout

6.7.2.4 card64 Coral::RTCPAbstractServer::Client::TimeStamp

6.7.2.5 void* Coral::RTCPAbstractServer::Client::UserData

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

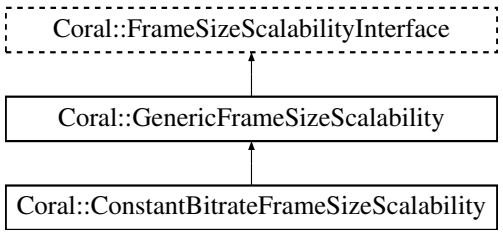
- rtcpabstractserver.h

6.8 Coral::ConstantBitrateFrameSizeScalability Class Reference

Constant Bitrate Frame Size Scalability.

`#include <cbrframesizescalability.h>`

Inheritance diagram for Coral::ConstantBitrateFrameSizeScalability:



Public Member Functions

- ConstantBitrateFrameSizeScalability ()
- ∼ConstantBitrateFrameSizeScalability ()
- void initConstantBitrateFrameSizeScalability (const cardinal maxFrameSize, const double scaleFactor)
- const char ∗ getFrameSizeScalabilityClass () const
- bool isFrameSizeScalable () const
- bool isVariableBitrate () const
- cardinal getMinPayloadFrameSizeForDelay (const double frameRate, const cardinal bufferDelay) const
- cardinal getMaxPayloadFrameSizeForDelay (const double frameRate, const cardinal bufferDelay) const
- cardinal getMaxFrameCountForDelay (const double frameRate, const cardinal bufferDelay) const
- cardinal getMaxBufferDelay (const double frameRate) const

Protected Attributes

- cardinal MinFrameSize
- cardinal MaxFrameSize

6.8.1 Detailed Description

Constant Bitrate Frame Size Scalability.

This class is an implementation of FrameSizeScalabilityInterface. Important node: All frames sizes in this class are payload frame sizes!

Author

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

6.8.2 Constructor & Destructor Documentation

6.8.2.1 Coral::ConstantBitrateFrameSizeScalability::ConstantBitrateFrameSizeScalability ( )

Constructor.

6.8.2.2 Coral::ConstantBitrateFrameSizeScalability::∼ConstantBitrateFrameSizeScalability ( )

Desstructor.

6.8.3 Member Function Documentation

6.8.3.1 const char * Coral::ConstantBitrateFrameSizeScalability::getFrameSizeScalabilityClass ( ) const [virtual]

Implementation of FrameSizeScalabilityInterface.

See also

FrameSizeScalabilityInterface::getFrameSizeScalabilityClass

Implements Coral::FrameSizeScalabilityInterface.

6.8.3.2 cardinal Coral::ConstantBitrateFrameSizeScalability::getMaxBufferDelay (const double *frameRate*) const [virtual]

Implementation of FrameSizeScalabilityInterface.

See also

FrameSizeScalabilityInterface::getMaxBufferDelay

Implements Coral::FrameSizeScalabilityInterface.

6.8.3.3 cardinal Coral::ConstantBitrateFrameSizeScalability::getMaxFrameCountForDelay (const double *frameRate,* const cardinal *bufferDelay*) const [virtual]

Implementation of FrameSizeScalabilityInterface.

See also

FrameSizeScalabilityInterface::getMaxFrameCountForDelay

Implements Coral::FrameSizeScalabilityInterface.

6.8.3.4 cardinal Coral::ConstantBitrateFrameSizeScalability::getMaxPayloadFrameSizeForDelay (const double *frameRate,* const cardinal *bufferDelay*) const [virtual]

Implementation of FrameSizeScalabilityInterface.

See also

FrameSizeScalabilityInterface::getMaxPayloadFrameSizeForDelay

Implements Coral::FrameSizeScalabilityInterface.

6.8.3.5 cardinal Coral::ConstantBitrateFrameSizeScalability::getMinPayloadFrameSizeForDelay (const double *frameRate,* const cardinal *bufferDelay*) const [virtual]

Implementation of FrameSizeScalabilityInterface.

See also

FrameSizeScalabilityInterface::getMinPayloadFrameSizeForDelay

Implements Coral::FrameSizeScalabilityInterface.

6.8.3.6 void Coral::ConstantBitrateFrameSizeScalability::initConstantBitrateFrameSizeScalability (const cardinal *maxFrameSize,* const double *scaleFactor*)

Initialize object with new maximum payload frame size and scale factor. MinFrameSize = scaleFactor ∗ MaxFrameSize.

Parameters

maxFrameSize	Maximum payload frame size.
scaleFactor	Scale factor.

6.8.3.7 bool Coral::ConstantBitrateFrameSizeScalability::isFrameSizeScalable ( ) const [virtual]

Implementation of FrameSizeScalabilityInterface.

See also

FrameSizeScalabilityInterface::isFrameSizeScalable

Implements Coral::FrameSizeScalabilityInterface.

6.8.3.8 bool Coral::ConstantBitrateFrameSizeScalability::isVariableBitrate ( ) const [virtual]

Implementation of FrameSizeScalabilityInterface.

See also

FrameSizeScalabilityInterface::isVariableBitRate

Implements Coral::FrameSizeScalabilityInterface.

6.8.4 Member Data Documentation

6.8.4.1 cardinal Coral::ConstantBitrateFrameSizeScalability::MaxFrameSize [protected]

6.8.4.2 cardinal Coral::ConstantBitrateFrameSizeScalability::MinFrameSize [protected]

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

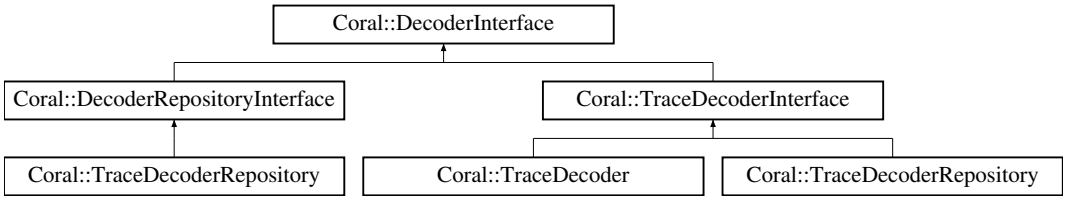
- cbrframesizescalability.h
- cbrframesizescalability.cc

6.9 Coral::DecoderInterface Class Reference

Decoder Interface.

`#include <decoderinterface.h>`

Inheritance diagram for Coral::DecoderInterface:



Public Member Functions

- virtual ~DecoderInterface ()
- 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

6.9.1 Detailed Description

Decoder Interface.

This class is the interface for a decoder.

Author

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

6.9.2 Constructor & Destructor Documentation

6.9.2.1 Coral::DecoderInterface::~DecoderInterface ( ) [virtual]

Virtual destructor.

6.9.3 Member Function Documentation

6.9.3.1 virtual void Coral::DecoderInterface::activate ( ) [pure virtual]

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

Implemented in Coral::TraceDecoderRepository, and Coral::TraceDecoder.

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

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

Parameters

decoder-Packet	DecoderPacket structure.

Returns

true, if packet is valid; false otherwise.

Implemented in Coral::TraceDecoderRepository, and Coral::TraceDecoder.

6.9.3.3 virtual void Coral::DecoderInterface::deactivate ( ) [pure virtual]

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

Implemented in Coral::TraceDecoderRepository, and Coral::TraceDecoder.

6.9.3.4 virtual card8 Coral::DecoderInterface::getErrorCode ( ) const [pure virtual]

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

Returns

Error code

Implemented in Coral::TraceDecoderRepository, and Coral::TraceDecoder.

6.9.3.5 virtual card64 Coral::DecoderInterface::getMaxPosition ( ) const [pure virtual]

Get maximum position in nanoseconds.

Returns

Maximum position in nanoseconds.

Implemented in Coral::TraceDecoderRepository, and Coral::TraceDecoder.

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

Get media info.

Parameters

mediaInfo	Reference to store MediaInfo to.

Implemented in Coral::TraceDecoderRepository, and Coral::TraceDecoder.

6.9.3.7 virtual card64 Coral::DecoderInterface::getPosition ( ) const [pure virtual]

Get current position in nanoseconds.

Returns

Position in nanoseconds.

Implemented in Coral::TraceDecoderRepository, and Coral::TraceDecoder.

6.9.3.8 virtual const card16 Coral::DecoderInterface::getTypeID ( ) const [pure virtual]

Get the decoder's type ID.

Returns

Decoder's type ID.

Implemented in Coral::TraceDecoderRepository, and Coral::TraceDecoder.

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

Get the decoder's name.

Returns

Decoder's name

Implemented in Coral::TraceDecoderRepository, and Coral::TraceDecoder.

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

Handle next received packet.

Parameters

decoder-Packet	DecoderPacket structure.

Implemented in Coral::TraceDecoderRepository, and Coral::TraceDecoder.

6.9.3.11 virtual void Coral::DecoderInterface::reset ( ) [pure virtual]

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

Implemented in Coral::TraceDecoderRepository, and Coral::TraceDecoder.

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

- decoderinterface.h
- decoderinterface.cc

6.10 Coral::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

6.10.1 Detailed Description

DecoderPacket.

This structure contains packet information for handleNextPacket() call.

Author

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

6.10.2 Member Data Documentation

6.10.2.1 void* Coral::DecoderPacket::Buffer

Buffer to write packet payload into.

6.10.2.2 cardinal Coral::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.

6.10.2.3 cardinal Coral::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.

6.10.2.4 cardinal Coral::DecoderPacket::Length

Maximum length of payload to be written into Buffer.

6.10.2.5 bool Coral::DecoderPacket::Marker

The packet's marker.

6.10.2.6 card8 Coral::DecoderPacket::PayloadType

The packet's payload type.

6.10.2.7 card16 Coral::DecoderPacket::SequenceNumber

The packet's sequence number.

6.10.2.8 SourceStateInfo** Coral::DecoderPacket::SSIArray

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

6.10.2.9 card32 Coral::DecoderPacket::TimeStamp

The packet's time stamp.

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

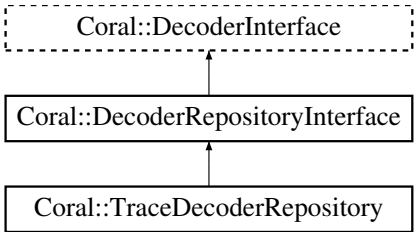
- decoderinterface.h

6.11 Coral::DecoderRepositoryInterface Class Reference

Decoder Repository.

`#include <decoderrepositoryinterface.h>`

Inheritance diagram for Coral::DecoderRepositoryInterface:



Public Member Functions

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

6.11.1 Detailed Description

Decoder Repository.

This class is a repository for decoders.

Author

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

6.11.2 Member Function Documentation

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

Get DecoderInterface of the current decoder.

Returns

Current decoder's DecoderInterface.

Implemented in Coral::TraceDecoderRepository.

6.11.2.2 virtual bool Coral::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 Coral::TraceDecoderRepository.

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

- decoderrepositoryinterface.h

6.12 Coral::DiffServClass Struct Reference

DiffServ Class.

`#include <servicelevelagreement.h>`

Public Attributes

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

6.12.1 Detailed Description

DiffServ Class.

This is a DiffServ class.

Author

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

6.12.2 Member Data Documentation

6.12.2.1 card64 Coral::DiffServClass::BytesPerSecond

Bytes per second.

6.12.2.2 double Coral::DiffServClass::CostFactor

Cost factor.

6.12.2.3 double Coral::DiffServClass::DelayVariability

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

6.12.2.4 double Coral::DiffServClass::MaxJitter

Maximum jitter.

6.12.2.5 double Coral::DiffServClass::MaxLossRate

Maximum loss rate.

6.12.2.6 double Coral::DiffServClass::MaxTransferDelay

Maximum transfer delay.

6.12.2.7 card8 Coral::DiffServClass::TrafficClass

Traffic class byte.

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

- servicelevelagreement.h

6.13 Coral::EmpiricalEnvelope Struct Reference

Empirical Envelope Header.

`#include <tdtf.h>`

Public Member Functions

- cardinal getConstraint (const cardinal bufferDelay) const

Static Public Member Functions

- static cardinal getSize (const cardinal eePairs)

Public Attributes

- card16 Pairs
- EmpiricalEnvelopePair Pair [0]

6.13.1 Detailed Description

Empirical Envelope Header.

This is the header for an empirical envelope.

Author

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

6.13.2 Member Function Documentation

6.13.2.1 cardinal Coral::EmpiricalEnvelope::getConstraint (const cardinal *bufferDelay*) const

Get constraint for given buffer delay.

Parameters

bufferDelay	Buffer delay.

Returns

Constraint.

6.13.2.2 static cardinal Coral::EmpiricalEnvelope::getSize (const cardinal *eePairs*) [inline, static]

Get empirical envelope size for given number of pairs.

Parameters

eePairs	Pairs.

Returns

Size.

6.13.3 Member Data Documentation

6.13.3.1 EmpiricalEnvelopePair Coral::EmpiricalEnvelope::Pair[0]

Array of D-BIND entries.

6.13.3.2 card16 Coral::EmpiricalEnvelope::Pairs

Number of D-BIND entries.

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

- tdtf.h
- tdtf.cc

6.14 Coral::EmpiricalEnvelopePair Struct Reference

Empirical Envelope Pair.

`#include <tdtf.h>`

Public Attributes

- card16 Interval
- card32 Sum

6.14.1 Detailed Description

Empirical Envelope Pair.

This is a D-BIND entry of an empirical envelope.

Author

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

6.14.2 Member Data Documentation

6.14.2.1 card16 Coral::EmpiricalEnvelopePair::Interval

Interval length.

6.14.2.2 card32 Coral::EmpiricalEnvelopePair::Sum

Sum of bytes, frames, etc.: E(t).

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

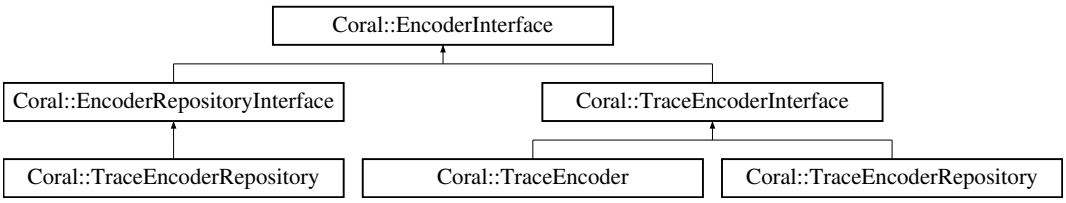
- tdtf.h

6.15 Coral::EncoderInterface Class Reference

Encoder Interface.

`#include <encoderinterface.h>`

Inheritance diagram for Coral::EncoderInterface:



Public Member Functions

- virtual ~EncoderInterface ()
- 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 AbstractQoSDescription * getQoSDescription (const cardinal pktHeaderSize, const cardinal pktMaxSize, const card64 offset)=0
- virtual void updateQuality (const AbstractQoSDescription *aqd)=0

6.15.1 Detailed Description

Encoder Interface.

This class is the interface for an encoder.

Author

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

6.15.2 Constructor & Destructor Documentation

6.15.2.1 Coral::EncoderInterface::~EncoderInterface ( ) [virtual]

Virtual destructor.

6.15.3 Member Function Documentation

6.15.3.1 virtual void Coral::EncoderInterface::activate ( ) [pure virtual]

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

Implemented in Coral::TraceEncoderRepository, and Coral::TraceEncoder.

6.15.3.2 virtual bool Coral::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 Coral::TraceEncoderRepository, and Coral::TraceEncoder.

6.15.3.3 virtual void Coral::EncoderInterface::deactivate ( ) [pure virtual]

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

Implemented in Coral::TraceEncoderRepository, and Coral::TraceEncoder.

6.15.3.4 virtual cardinal Coral::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 Coral::TraceEncoderRepository, and Coral::TraceEncoder.

6.15.3.5 virtual AbstractQoSDescription* Coral::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 Coral::TraceEncoderRepository, and Coral::TraceEncoder.

6.15.3.6 virtual const card16 Coral::EncoderInterface::getTypeID ( ) const [pure virtual]

Get the encoder's type ID.

Returns

Encoder's type ID.

Implemented in Coral::TraceEncoderRepository, and Coral::TraceEncoder.

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

Get the encoder's name.

Returns

Encoder's name

Implemented in Coral::TraceEncoderRepository, and Coral::TraceEncoder.

6.15.3.8 virtual bool Coral::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)
maxPacket-Size	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 Coral::TraceEncoderRepository, and Coral::TraceEncoder.

6.15.3.9 virtual void Coral::EncoderInterface::reset ( ) [pure virtual]

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

Implemented in Coral::TraceEncoderRepository, and Coral::TraceEncoder.

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

Update encoder quality to changes made in QoS description returned by getQoS-Description().

See also

EncoderInterface::getQoSDescription

Implemented in Coral::TraceEncoderRepository, and Coral::TraceEncoder.

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

- encoderinterface.h
- encoderinterface.cc

6.16 Coral::EncoderPacket Struct Reference

EncoderPacket.

```
#include <encoderinterface.h>
```

Public Attributes

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

6.16.1 Detailed Description

EncoderPacket.

This structure contains packet information for getNextPacket() call.

Author

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

6.16.2 Member Data Documentation

6.16.2.1 void* Coral::EncoderPacket::Buffer

Buffer to write packet payload into.

6.16.2.2 card8 Coral::EncoderPacket::ErrorCode

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

6.16.2.3 cardinal Coral::EncoderPacket::Layer

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

6.16.2.4 bool Coral::EncoderPacket::Marker

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

6.16.2.5 cardinal Coral::EncoderPacket::MaxLength

Maximum length of payload to be written into Buffer.

6.16.2.6 card8 Coral::EncoderPacket::PayloadType

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

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

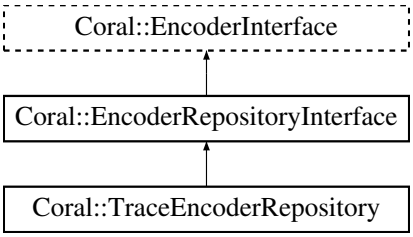
- encoderinterface.h

6.17 Coral::EncoderRepositoryInterface Class Reference

Encoder Repository Interface.

`#include <encoderrepositoryinterface.h>`

Inheritance diagram for Coral::EncoderRepositoryInterface:



Public Member Functions

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

6.17.1 Detailed Description

Encoder Repository Interface.

This class is a repository for encoders.

Author

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

6.17.2 Member Function Documentation

6.17.2.1 virtual EncoderInterface* Coral::EncoderRepository-Interface::getCurrentEncoder ( ) const [pure virtual]

Get EncoderInterface of the current encoder.

Returns

Current encoder's EncoderInterface.

Implemented in Coral::TraceEncoderRepository.

6.17.2.2 virtual bool Coral::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 Coral::TraceEncoderRepository.

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

- encoderrepositoryinterface.h

6.18 Coral::FrameDescription Struct Reference

Frame Description.

`#include <tdtf.h>`

Public Attributes

- card32 ID
- card32 Size

6.18.1 Detailed Description

Frame Description.

This is an entry of a trace, it describes one frame.

Author

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

6.18.2 Member Data Documentation

6.18.2.1 card32 Coral::FrameDescription::ID

ID of the frame.

6.18.2.2 card32 Coral::FrameDescription::Size

Size of the frame.

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

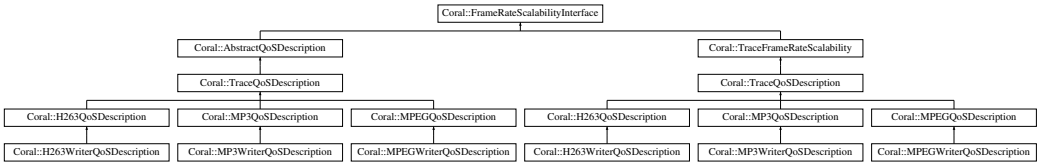
- tdtf.h

6.19 Coral::FrameRateScalabilityInterface Class Reference

Frame Rate Scalability Interface.

`#include <frameratescalabilityinterface.h>`

Inheritance diagram for Coral::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

6.19.1 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

6.19.2 Member Function Documentation

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

Get name of the frame rate scalability class.

Returns

Frame rate scalability class name.

Implemented in Coral::TraceQoSDescription, and Coral::TraceFrameRateScalability.

6.19.2.2 virtual double Coral::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])..

Implemented in Coral::TraceFrameRateScalability.

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

Get utilization for given frame rate.

Parameters

frameRate	Frame rate.

Returns

Utilization (out of [0,1]).

Implemented in Coral::TraceFrameRateScalability.

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

Get frame rate utilization weight.

Parameters

frameRate	Frame rate.

Returns

Utilization weight.

Implemented in Coral::TraceFrameRateScalability.

6.19.2.5 virtual double Coral::FrameRateScalabilityInterface::getMaxFrameRate ( ) const [pure virtual]

Get maximum frame rate.

Returns

Maximum frame rate.

Implemented in Coral::TraceFrameRateScalability.

6.19.2.6 virtual double Coral::FrameRateScalabilityInterface::getMinFrameRate ( ) const [pure virtual]

Get minimum frame rate.

Returns

Minimum frame rate.

Implemented in Coral::TraceFrameRateScalability.

6.19.2.7 virtual double Coral::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.

Implemented in Coral::TraceFrameRateScalability.

6.19.2.8 virtual double Coral::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.

Implemented in Coral::TraceFrameRateScalability.

6.19.2.9 virtual double Coral::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.

Implemented in Coral::TraceFrameRateScalability.

6.19.2.10 virtual bool Coral::FrameRateScalabilityInterface::isFrameRateScalable ( ) const [pure virtual]

Check, if frame rate is scalable.

Returns

true, if frame rate is scalable; false otherwise.

Implemented in Coral::TraceFrameRateScalability.

6.19.2.11 virtual bool Coral::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.

Implemented in Coral::TraceFrameRateScalability.

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

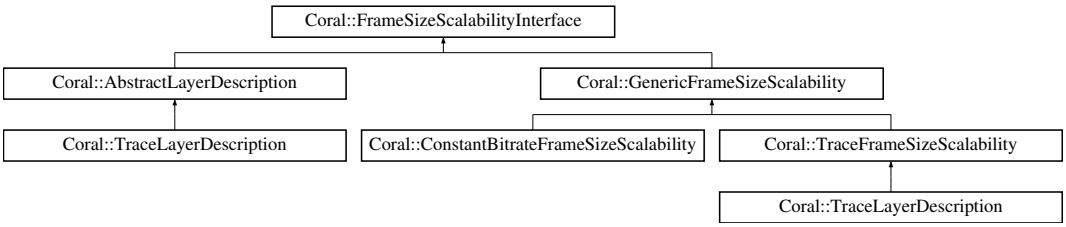
- frameratescalabilityinterface.h

6.20 Coral::FrameSizeScalabilityInterface Class Reference

Frame Rate Scalability Interface.

`#include <framesizescalabilityinterface.h>`

Inheritance diagram for Coral::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

6.20.1 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

6.20.2 Member Function Documentation

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

Get name of the frame size scalability class.

Returns

Frame size scalability class name.

Implemented in Coral::TraceFrameSizeScalability, Coral::ConstantBitrateFrameSizeScalability, and Coral::TraceLayerDescription.

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

Get frame size utilization weight.

Parameters

frameRate	Frame rate.

Returns

Utilization weight.

Implemented in Coral::TraceFrameSizeScalability, and Coral::GenericFrameSizeScalability.

6.20.2.3 virtual cardinal Coral::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.

Implemented in Coral::TraceFrameSizeScalability, and Coral::ConstantBitrateFrameSizeScalability.

6.20.2.4 virtual cardinal Coral::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.

Implemented in Coral::TraceFrameSizeScalability, and Coral::ConstantBitrateFrameSizeScalability.

6.20.2.5 virtual cardinal Coral::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.

Implemented in Coral::TraceFrameSizeScalability, and Coral::ConstantBitrateFrameSizeScalability.

6.20.2.6 virtual cardinal Coral::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.

Implemented in Coral::TraceFrameSizeScalability, and Coral::ConstantBitrateFrameSizeScalability.

6.20.2.7 virtual cardinal Coral::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.

Implemented in Coral::GenericFrameSizeScalability.

6.20.2.8 virtual cardinal Coral::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.

Implemented in Coral::GenericFrameSizeScalability.

6.20.2.9 virtual cardinal Coral::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.

Implemented in Coral::GenericFrameSizeScalability.

6.20.2.10 virtual double Coral::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])..

Implemented in Coral::GenericFrameSizeScalability.

6.20.2.11 virtual double Coral::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])..

Implemented in Coral::TraceFrameSizeScalability, and Coral::GenericFrameSizeScalability.

6.20.2.12 virtual cardinal Coral::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.

Implemented in Coral::GenericFrameSizeScalability.

6.20.2.13 virtual cardinal Coral::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.

Implemented in Coral::GenericFrameSizeScalability.

6.20.2.14 virtual bool Coral::FrameSizeScalabilityInterface::isFrameSizeScalable ( ) const [pure virtual]

Check, if frame size is scalable.

Returns

true, if frame size is scalable; false otherwise.

Implemented in Coral::TraceFrameSizeScalability, and Coral::ConstantBitrateFrameSizeScalability.

6.20.2.15 virtual bool Coral::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.

Implemented in Coral::GenericFrameSizeScalability.

6.20.2.16 virtual bool Coral::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.

Implemented in Coral::TraceFrameSizeScalability, and Coral::ConstantBitrateFrameSizeScalability.

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

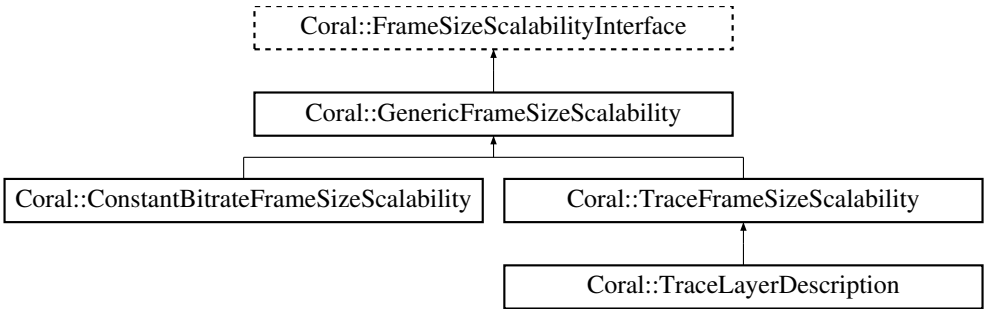
- framesizescalabilityinterface.h

6.21 Coral::GenericFrameSizeScalability Class Reference

Generic Frame Size Scalability.

`#include <genericframesizescalability.h>`

Inheritance diagram for Coral::GenericFrameSizeScalability:



Public Member Functions

- bool isValidPayloadFrameSize (const double frameRate, const cardinal bufferDelay, const cardinal frameSize) const
- cardinal getNearestValidPayloadFrameSize (const double frameRate, const cardinal bufferDelay, const cardinal frameSize) const
- cardinal getNextPayloadFrameSizeForDelayAndSize (const double frameRate, const cardinal bufferDelay, const cardinal frameSize) const
- cardinal getPrevPayloadFrameSizeForDelayAndSize (const double frameRate, const cardinal bufferDelay, const cardinal frameSize) const
- double getPayloadFrameSizeScaleFactorForDelayAndSize (const double frameRate, const cardinal bufferDelay, const cardinal frameSize) const
- double getPayloadFrameSizeUtilizationForDelayAndSize (const double frameRate, const cardinal bufferDelay, const cardinal frameSize) const
- double getFrameSizeUtilizationWeight (const double frameRate) const
- cardinal getNextBufferDelayForDelay (const double frameRate, const cardinal bufferDelay) const
- cardinal getPrevBufferDelayForDelay (const double frameRate, const cardinal bufferDelay) const

6.21.1 Detailed Description

Generic Frame Size Scalability.

This class is a generic implementation of FrameSizeScalabilityInterface. It provides basic functionality for subclasses. Important node: All frames sizes in this class are payload frame sizes!

Author

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

6.21.2 Member Function Documentation

6.21.2.1 double Coral::GenericFrameSizeScalability::getFrameSizeUtilizationWeight (const double *frameRate*) const [virtual]

Implementation of FrameSizeScalabilityInterface.

See also

FrameSizeScalabilityInterface::getFrameSizeUtilizationWeight

Implements Coral::FrameSizeScalabilityInterface.

Reimplemented in Coral::TraceFrameSizeScalability.

6.21.2.2 cardinal Coral::GenericFrameSizeScalability::getNearestValidPayloadFrameSize (const double *frameRate,* const cardinal *bufferDelay,* const cardinal *frameSize*) const [virtual]

Implementation of FrameSizeScalabilityInterface.

See also

FrameSizeScalabilityInterface::getNearestValidPayloadFrameSize

Implements Coral::FrameSizeScalabilityInterface.

6.21.2.3 cardinal Coral::GenericFrameSizeScalability::getNextBufferDelayForDelay (const double *frameRate,* const cardinal *bufferDelay*) const [virtual]

Implementation of FrameSizeScalabilityInterface.

See also

FrameSizeScalabilityInterface::getNextBufferDelayForDelay

Implements Coral::FrameSizeScalabilityInterface.

6.21.2.4 cardinal Coral::GenericFrameSizeScalability::getNextPayloadFrameSizeForDelayAndSize (const double *frameRate,* const cardinal *bufferDelay,* const cardinal *frameSize*) const [virtual]

Implementation of FrameSizeScalabilityInterface.

See also

FrameSizeScalabilityInterface::getNextFrameSizeForDelayAndSize

Implements Coral::FrameSizeScalabilityInterface.

6.21.2.5 double Coral::GenericFrameSizeScalability::getPayloadFrameSizeScaleFactorForDelayAndSize (const double *frameRate,* const cardinal *bufferDelay,* const cardinal *frameSize*) const [virtual]

Implementation of FrameSizeScalabilityInterface.

See also

FrameSizeScalabilityInterface::getPayloadFrameSizeScaleFactorForDelayAndSize

Implements Coral::FrameSizeScalabilityInterface.

6.21.2.6 double Coral::GenericFrameSizeScalability::getPayloadFrameSizeUtilizationForDelayAndSize (const double *frameRate,* const cardinal *bufferDelay,* const cardinal *frameSize*) const [virtual]

Implementation of FrameSizeScalabilityInterface.

See also

FrameSizeScalabilityInterface::getPayloadFrameSizeUtilizationForDelayAndSize

Implements Coral::FrameSizeScalabilityInterface.

Reimplemented in Coral::TraceFrameSizeScalability.

6.21.2.7 cardinal Coral::GenericFrameSizeScalability::getPrevBufferDelayForDelay (const double *frameRate,* const cardinal *bufferDelay*) const [virtual]

Implementation of FrameSizeScalabilityInterface.

See also

FrameSizeScalabilityInterface::getPrevBufferDelayForDelay

Implements Coral::FrameSizeScalabilityInterface.

6.21.2.8 cardinal Coral::GenericFrameSizeScalability::getPrevPayloadFrameSizeForDelayAndSize (const double *frameRate,* const cardinal *bufferDelay,* const cardinal *frameSize*) const [virtual]

Implementation of FrameSizeScalabilityInterface.

See also

FrameSizeScalabilityInterface::getPrevFrameSizeForDelayAndSize

Implements Coral::FrameSizeScalabilityInterface.

6.21.2.9 bool Coral::GenericFrameSizeScalability::isValidPayloadFrameSize (const double *frameRate*, const cardinal *bufferDelay*, const cardinal *frameSize*) const [virtual]

Implementation of FrameSizeScalabilityInterface.

See also

FrameSizeScalabilityInterface::isValidPayloadFrameSize

Implements Coral::FrameSizeScalabilityInterface.

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

- genericframesizescalability.h
- genericframesizescalability.cc

6.22 Coral::GNUPlotData Class Reference

GNUPlot Data.

`#include <gnuplot.h>`

Public Member Functions

- GNUPlotData ()
- ~GNUPlotData ()
- bool open (const char *prefix)
- void close ()

Public Attributes

- bool Ready
- String Name
- ofstream Stream

6.22.1 Detailed Description

GNUPlot Data.

This is a GNUplot data writer class.

Author

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

6.22.2 Constructor & Destructor Documentation

6.22.2.1 Coral::GNUPlotData::GNUPlotData ( )

Constructor.

6.22.2.2 Coral::GNUPlotData::~GNUPlotData ( )

Destructor.

6.22.3 Member Function Documentation

6.22.3.1 void Coral::GNUPlotData::close ( )

Close data file.

6.22.3.2 bool Coral::GNUPlotData::open (const char * *prefix*)

Open data file. The extension ".data" will be added automatically.

Parameters

prefix	Name prefix.

Returns

true, if file has been opened; false otherwise.

6.22.4 Member Data Documentation

6.22.4.1 String Coral::GNUPlotData::Name

Name of the output file.

6.22.4.2 bool Coral::GNUPlotData::Ready

True, if output stream is opened; false otherwise.

6.22.4.3 ofstream Coral::GNUPlotData::Stream

Output stream.

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

- gnuplot.h
- gnuplot.cc

6.23 Coral::GNUPlotScript Class Reference

GNUPlot Data.

`#include <gnuplot.h>`

Public Member Functions

- GNUPlotScript ()
- ~GNUPlotScript ()
- bool open (const char *prefix)
- void close ()
- void plot (const char *pagetitle, const char *linetitle, const char *xlabel, const char *ylabel, const char *info, const bool ownPage, const GNUPlotData &data, const cardinal column, const int64 xRangeMin=0, const int64 xRangeMax=0, const int64 yRangeMin=0, const int64 yRangeMax=0)
- void done ()

Private Member Functions

- void newPage (const char *title, const char *xlabel, const char *ylabel, const char *info)

Private Attributes

- cardinal Usage
- String Name
- ofstream Stream
- card64 TimeStamp
- String InfoString
- bool Ready

6.23.1 Detailed Description

GNUPlot Data.

This is a GNUplot script writer class.

Author

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

6.23.2 Constructor & Destructor Documentation

6.23.2.1 Coral::GNUPlotScript::GNUPlotScript ( )

Constructor.

6.23.2.2 Coral::GNUPlotScript::~GNUPlotScript ( )

Destructor.

6.23.3 Member Function Documentation

6.23.3.1 void Coral::GNUPlotScript::close ( )

Close script file.

6.23.3.2 void Coral::GNUPlotScript::done ( )

End multi-line plot command.

6.23.3.3 void Coral::GNUPlotScript::newPage (const char * *title,* const char * *xlabel,* const char * *ylabel,* const char * *info*) [private]

6.23.3.4 bool Coral::GNUPlotScript::open (const char * *prefix*)

Open script file. The extension ".gp" will be added automatically.

Parameters

prefix	Name prefix.

Returns

true, if file has been opened; false otherwise.

6.23.3.5 void Coral::GNUPlotScript::plot (const char * *pagetitle,* const char * *linetitle,* const char * *xlabel,* const char * *ylabel,* const char * *info,* const bool *ownPage,* const GNUPlotData & *data,* const cardinal *column,* const int64 *xRangeMin* = 0, const int64 *xRangeMax* = 0, const int64 *yRangeMin* = 0, const int64 *yRangeMax* = 0)

Write "plot" command.

Parameters

pagetitle	Page title.
linetitle	Line title.
xlabel	X axis label.
ylabel	Y axis label.
info	Information string.
ownPage	true, to begin own page for this plot; false otherwise.
data	GNUPlotData containing data to plot. üram column Column of data to plot.
xRangeMin	Minimum X value; 0 = Auto-scale.
xRangeMax	Maximum X value; 0 = Auto-scale.
yRangeMin	Minimum Y value; 0 = Auto-scale.
yRangeMax	Maximum Y value; 0 = Auto-scale.

6.23.4 Member Data Documentation

6.23.4.1 String Coral::GNUPlotScript::InfoString [private]

6.23.4.2 String Coral::GNUPlotScript::Name [private]

6.23.4.3 bool Coral::GNUPlotScript::Ready [private]

6.23.4.4 ofstream Coral::GNUPlotScript::Stream [private]

6.23.4.5 card64 Coral::GNUPlotScript::TimeStamp [private]

6.23.4.6 cardinal Coral::GNUPlotScript::Usage [private]

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

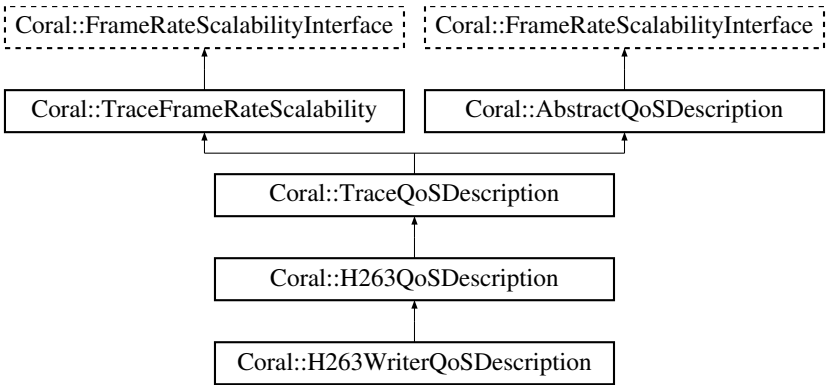
- gnuplot.h
- gnuplot.cc

6.24 Coral::H263QoSDescription Class Reference

H263 QoS Description.

`#include <h263qosdescription.h>`

Inheritance diagram for Coral::H263QoSDescription:



Public Member Functions

- void calculateMaxUtilizationForBandwidthArray (const card64 *totalBandwidthArray, ResourceUtilizationPoint *rupArray, const cardinal points) const

Private Member Functions

- bool tryAllocation (card64 &available, card64 *requiredArray, card64 *bandwidthArray, cardinal layers) const

6.24.1 Detailed Description

H263 QoS Description.

This class extends TraceWriterQoSDescription by H263-specific utilization calculation necessary to generate resource/utilization lists.

Author

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

6.24.2 Member Function Documentation

6.24.2.1 void Coral::H263QoSDescription::calculateMaxUtilizationForBandwidthArray (const card64 * *totalBandwidthArray*, ResourceUtilizationPoint * *rupArray*, const cardinal *points*) const [virtual]

Implementation of AbstractQoSDescription's calculateUtilizationForLayerBandwidths().

See also

AbstractQoSDescription::calculateUtilizationForLayerBandwidths

Reimplemented from Coral::AbstractQoSDescription.

6.24.2.2 bool Coral::H263QoSDescription::tryAllocation (card64 & *available*, card64 * *requiredArray*, card64 * *bandwidthArray*, cardinal *layers*) const [private]

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

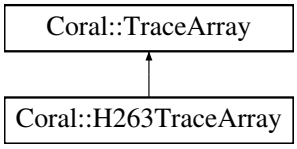
- h263qosdescription.h
- h263qosdescription.cc

6.25 Coral::H263TraceArray Class Reference

H263 Trace Array.

`#include <h263tracearray.h>`

Inheritance diagram for Coral::H263TraceArray:



Public Member Functions

- H263TraceArray (const TraceConfiguration &config)
- bool load (const char *name, const cardinal frameRate, const cardinal extLayers, const double fakeE1, const double fakeE2)
- TraceArray * H263TraceArray::decreaseFrameRate () const

Static Public Attributes

- static const cardinal LayerH263BaseI = 0
- static const cardinal LayerH263BaseP = 1
- static const cardinal LayerH263BasePB = 2
- static const cardinal LayerH263BaseB = 3
- static const cardinal LayerH263ExtI1 = 4
- static const cardinal LayerH263ExtP1 = 5
- static const cardinal LayerH263ExtPB1 = 6
- static const cardinal LayerH263ExtB1 = 7

- static const cardinal LayerH263ExtI2 = 8
- static const cardinal LayerH263ExtP2 = 9
- static const cardinal LayerH263ExtPB2 = 10
- static const cardinal LayerH263ExtB2 = 11

6.25.1 Detailed Description

H263 Trace Array.

This is an array of H263 layer traces.

Author

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

6.25.2 Constructor & Destructor Documentation

6.25.2.1 Coral::H263TraceArray::H263TraceArray (const TraceConfiguration & *config*)

Constructor.

6.25.3 Member Function Documentation

6.25.3.1 TraceArray* Coral::H263TraceArray::H263TraceArray::decreaseFrameRate ( ) const

Reimplementation of TraceArray's decreaseFrameRate() method.

See also

TraceArray::decreaseFrameRate

6.25.3.2 bool Coral::H263TraceArray::load (const char * *name*, const cardinal *frameRate*, const cardinal *extLayers*, const double *fakeE1*, const double *fakeE2*)

Load H263 trace.

Parameters

name	File name.
frameRate	Frame rate of the trace file.
extLayers	Number of extension layers (0, 1 or 2).
fakeE1	Fake 1st extension layer: FrameSizeE1 = fakeE1 * FrameSizeBase.
fakeE2	Fake 2nd extension layer: FrameSizeE2 = fakeE2 * FrameSizeBase.

Returns

true, if load has been successful; false otherwise.

6.25.4 Member Data Documentation

6.25.4.1 const cardinal Coral::H263TraceArray::LayerH263BaseB = 3 [static]

H263-I/II B-frames layer.

6.25.4.2 const cardinal Coral::H263TraceArray::LayerH263BaseI = 0 [static]

H263-I/II I-frames layer.

6.25.4.3 const cardinal Coral::H263TraceArray::LayerH263BaseP = 1 [static]

H263-I/II P-frames layer.

6.25.4.4 const cardinal Coral::H263TraceArray::LayerH263BasePB = 2 [static]

H263-I/II PB-frames layer.

6.25.4.5 const cardinal Coral::H263TraceArray::LayerH263ExtB1 = 7 [static]

H263-I/II B-frames layer.

6.25.4.6 const cardinal Coral::H263TraceArray::LayerH263ExtB2 = 11 [static]

H263-I/II B-frames layer.

6.25.4.7 const cardinal Coral::H263TraceArray::LayerH263ExtI1 = 4 [static]

H263-I/II I-frames layer.

6.25.4.8 const cardinal Coral::H263TraceArray::LayerH263ExtI2 = 8 [static]

H263-I/II I-frames layer.

6.25.4.9 const cardinal Coral::H263TraceArray::LayerH263ExtP1 = 5 [static]

H263-I/II P-frames layer.

6.25.4.10 const cardinal Coral::H263TraceArray::LayerH263ExtP2 = 9 [static]

H263-I/II P-frames layer.

6.25.4.11 const cardinal Coral::H263TraceArray::LayerH263ExtPB1 = 6 [static]

H263-I/II PB-frames layer.

6.25.4.12 const cardinal Coral::H263TraceArray::LayerH263ExtPB2 = 10 [static]

H263-I/II PB-frames layer.

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

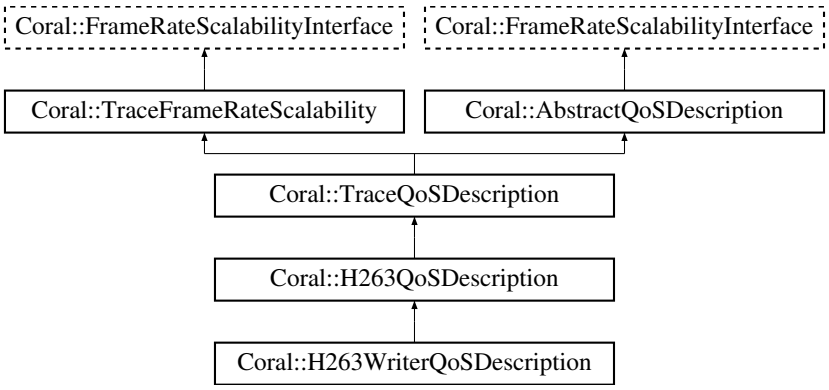
- h263tracearray.h
- h263tracearray.cc

6.26 Coral::H263WriterQoSDescription Class Reference

H263 Writer QoS Description.

`#include <h263writerqosdescription.h>`

Inheritance diagram for Coral::H263WriterQoSDescription:



Public Member Functions

- H263WriterQoSDescription (const TraceConfiguration *traceConfiguration)
- double calculateUtilizationForLayerBandwidths (const double frameRate, const cardinal layers, const cardinal *bandwidth) const

Private Attributes

- const TraceConfiguration * Config

6.26.1 Detailed Description

H263 Writer QoS Description.

This is the QoS description of a trace, extended by methods for H263 resource/utilization calculation. These overwrite the ones which are used for calculating utilization from the trace file by the ones which use a new configuration. This is necessary to update the utilization of the file.

Author

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

6.26.2 Constructor & Destructor Documentation

6.26.2.1 Coral::H263WriterQoSDescription::H263WriterQoSDescription (const TraceConfiguration * *traceConfiguration*)

Constructor.

Parameters

trace-Configuration	Configuration.

6.26.3 Member Function Documentation

6.26.3.1 double Coral::H263WriterQoSDescription::calculateUtilizationForLayer-Bandwidths (const double *frameRate,* const cardinal *layers,* const cardinal * *bandwidth*) const

Implementation of AbstractQoSDescription's calculateUtilizationForLayerBandwidths().

See also

AbstractQoSDescription::calculateUtilizationForLayerBandwidths

6.26.4 Member Data Documentation

6.26.4.1 const TraceConfiguration* Coral::H263WriterQoSDescription::Config [private]

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

- h263writerqosdescription.h
- h263writerqosdescription.cc

6.27 Coral::icmp_filter Struct Reference

Public Attributes

- __u32 data

6.27.1 Member Data Documentation

6.27.1.1 __u32 Coral::icmp_filter::data

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

- roundtriptimepinger.cc

6.28 in6_flowlabel_req Struct Reference

`#include <in6.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

6.28.1 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!

6.28.2 Member Data Documentation

6.28.2.1 __u32 in6_flowlabel_req::__flr_pad

6.28.2.2 __u8 in6_flowlabel_req::flr_action

6.28.2.3 struct in6_addr in6_flowlabel_req::flr_dst

6.28.2.4 __u16 in6_flowlabel_req::flr_expires

6.28.2.5 __u16 in6_flowlabel_req::flr_flags

6.28.2.6 __u32 in6_flowlabel_req::flr_label

6.28.2.7 __u16 in6_flowlabel_req::flr_linger

6.28.2.8 __u8 in6_flowlabel_req::flr_share

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

- in6.h

6.29 Coral::in6_flowlabel_req Struct Reference

```
#include <internetflow.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

6.29.1 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!

6.29.2 Member Data Documentation

6.29.2.1 __u32 Coral::in6_flowlabel_req::__flr_pad

6.29.2.2 __u8 Coral::in6_flowlabel_req::flr_action

6.29.2.3 struct in6_addr Coral::in6_flowlabel_req::flr_dst

6.29.2.4 __u16 Coral::in6_flowlabel_req::flr_expires

6.29.2.5 __u16 Coral::in6_flowlabel_req::flr_flags

6.29.2.6 __u32 Coral::in6_flowlabel_req::flr_label

6.29.2.7 __u16 Coral::in6_flowlabel_req::flr_linger

6.29.2.8 __u8 Coral::in6_flowlabel_req::flr_share

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

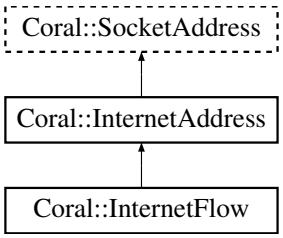
- in6.h

6.30 Coral::InternetAddress Class Reference

Socket Address.

`#include <internetaddress.h>`

Inheritance diagram for Coral::InternetAddress:



Public Types

- enum PrintFormat { PF_Address = (1 << 0), PF_Hostname = (1 << 1), PF_Full = PF_Address | PF_Hostname }

Public Member Functions

- InternetAddress ()

- InternetAddress (const InternetAddress &address)
- InternetAddress (const String &address)
- InternetAddress (const String &hostName, const card16 port)
- InternetAddress (const card16 port)
- InternetAddress (sockaddr *address, socklen_t length)
- ~InternetAddress ()
- void reset ()
- void init (const InternetAddress &address)
- void init (const card16 port)
- void init (const String &hostName, const card16 port)
- InternetAddress & operator= (const InternetAddress &source)
- card16 getPort () const
- void setPort (const card16 port)
- PrintFormat getPrintFormat () const
- void setPrintFormat (const PrintFormat format)
- bool isValid () const
- bool isNull () const
- String getAddressString () const
- bool isIPv4 () const
- bool isIPv6 () const
- cardinal getSystemAddress (sockaddr *buffer, const socklen_t length, const cardinal type) const
- bool setSystemAddress (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 InternetAddress getLocalAddress (const InternetAddress &peer)
- static cardinal getHostByName (const String &name, card16 *myadr)
- static bool hasIPv6 ()

Static Public Attributes

- static bool UseIPv6 = checkIPv6()

Static Private Member Functions

- static bool checkIPv6 ()

Private Attributes

- card16 Host [8]
- card16 Port
- PrintFormat Format
- bool Valid

6.30.1 Detailed Description

Socket Address.

This class manages an internet address.

Author

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

6.30.2 Member Enumeration Documentation

6.30.2.1 enum Coral::InternetAddress::PrintFormat

setPrintFormat() printing formats.

Enumerator:

PF_Address

PF_Hostname

PF_Full

6.30.3 Constructor & Destructor Documentation

6.30.3.1 Coral::InternetAddress::InternetAddress ( )

Constructor for an empty internet address.

6.30.3.2 Coral::InternetAddress::InternetAddress (const InternetAddress & *address*)

Constructor for an internet address from an internet address.

Parameters

address	Internet address.

6.30.3.3 Coral::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.

6.30.3.4 Coral::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.

6.30.3.5 Coral::InternetAddress::InternetAddress (const card16 *port*)

Constructor for INADDR_ANY address with given port.

Parameters

port	Port number.

6.30.3.6 Coral::InternetAddress::InternetAddress (sockaddr * *address,* 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)).

6.30.3.7 Coral::InternetAddress::~InternetAddress ( )

Destructor.

6.30.3.8 Coral::InternetAddress::InternetAddress (const PortableAddress & *address*)

Constructor for InternetAddress from PortableAddress.

Parameters

address	PortableAddress.

6.30.4 Member Function Documentation

6.30.4.1 bool Coral::InternetAddress::checkIPv6 ( ) [static, private]

6.30.4.2 String Coral::InternetAddress::getAddressString ( ) const [virtual]

Get address string.

Returns

Address string.

Implements Coral::SocketAddress.

Reimplemented in Coral::InternetFlow.

6.30.4.3 cardinal Coral::InternetAddress::getHostByName (const String & *name*, card16 * *myadr*) [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).
length	Length of the address saved in myaddr or 0 in case of failure.

6.30.4.4 InternetAddress Coral::InternetAddress::getLocalAddress (const InternetAddress & *peer*) [static]

Get the local host address. The parameter peer gives the address of the other host.

Parameters

address	Reference to SocketAddress to write address to.
peer	Address of peer.

Returns

true, if call was successful; false otherwise.

Examples: localhost => localhost address (127.0.0.1 or ::1). ethernet-host => ethernet interface address. internet-address => dynamic-ip address set by pppd.

6.30.4.5 card16 Coral::InternetAddress::getPort ( ) const [inline]

Get port of address.

6.30.4.6 PortableAddress Coral::InternetAddress::getPortableAddress ( ) const [inline]

Get PortableAddress from InternetAddress.

Returns

PortableAddress.

6.30.4.7 PrintFormat Coral::InternetAddress::getPrintFormat ( ) const [inline]

Get printing format.

Returns

Print format.

6.30.4.8 cardinal Coral::InternetAddress::getSystemAddress (sockaddr * *buffer*, const socklen_t *length*, const cardinal *type*) const [virtual]

getSystemAddress() implementation of SocketAddress

See also

SocketAddress::getSystemAddress

Implements Coral::SocketAddress.

Reimplemented in Coral::InternetFlow.

6.30.4.9 static bool Coral::InternetAddress::hasIPv6 ( ) [inline, static]

Check, if IPv6 support is available.

Returns

true, if IPv6 support is available; false otherwise.

6.30.4.10 void Coral::InternetAddress::init (const InternetAddress & *address*)

Initialize internet address from internet address.

6.30.4.11 void Coral::InternetAddress::init (const card16 *port*)

Initialize internet address with INADDR_ANY and given port.

Parameters

port	Port number.

6.30.4.12 void Coral::InternetAddress::init (const String & *hostName*, const card16 *port*)

Initialize internet address with given host name and port.

Parameters

hostName	Host name.
port	Port number.

6.30.4.13 void Coral::InternetAddress::init (const PortableAddress & *address*)

Initialize InternetAddress from PortableAddress.

Parameters

address	PortableAddress.

6.30.4.14 bool Coral::InternetAddress::isIPv4 ( ) const [inline]

Check, if internet address is an IPv4 or IPv4-mapped address.

Returns

true, if address is an IPv4 or IPv4-mapped address; false otherwise.

6.30.4.15 bool Coral::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.

6.30.4.16 bool Coral::InternetAddress::isNull ( ) const [inline]

Check, if the address is null.

Returns

true, if the address is not null; false otherwise.

6.30.4.17 bool Coral::InternetAddress::isValid ( ) const [virtual]

isValid() implementation of SocketAddress.

See also

SocketAddress::isValid

Implements Coral::SocketAddress.

6.30.4.18 int Coral::InternetAddress::operator!= (const InternetAddress & *address*) const [inline]

Implementation of != operator.

6.30.4.19 int Coral::InternetAddress::operator< (const InternetAddress & *address*) const

Implementation of < operator.

6.30.4.20 int Coral::InternetAddress::operator<= (const InternetAddress & *address*) const [inline]

Implementation of <= operator.

6.30.4.21 InternetAddress& Coral::InternetAddress::operator= (const InternetAddress & *source*) [inline]

Implementation of = operator.

6.30.4.22 int Coral::InternetAddress::operator== (const InternetAddress & *address*) const

Implementation of == operator.

6.30.4.23 int Coral::InternetAddress::operator> (const InternetAddress & *address*) const

Implementation of > operator.

6.30.4.24 int Coral::InternetAddress::operator>= (const InternetAddress & *address*) const [inline]

Implementation of >= operator.

6.30.4.25 void Coral::InternetAddress::reset ( ) [virtual]

Reset internet address.

Implements Coral::SocketAddress.

Reimplemented in Coral::InternetFlow.

6.30.4.26 void Coral::InternetAddress::setPort (const card16 *port*) [inline]

Set port of address.

6.30.4.27 void Coral::InternetAddress::setPrintFormat (const PrintFormat *format*) [inline]

Set printing format.

Parameters

format	Print format.

6.30.4.28 bool Coral::InternetAddress::setSystemAddress (sockaddr * *address*, const socklen_t *length*) [virtual]

setSystemAddress() implementation of SocketAddress.

See also

SocketAddress::setSystemAddress

Implements Coral::SocketAddress.

Reimplemented in Coral::InternetFlow.

6.30.5 Member Data Documentation

6.30.5.1 PrintFormat Coral::InternetAddress::Format [private]

Print format.

6.30.5.2 card16 Coral::InternetAddress::Host[8] [private]

Host address in network byte order. IPv4 addresses are converted to IPv4-mapped IPv6 addresses.

6.30.5.3 card16 Coral::InternetAddress::Port [private]

Port number.

6.30.5.4 bool Coral::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!!!

6.30.5.5 bool Coral::InternetAddress::Valid [private]

Is address valid?

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

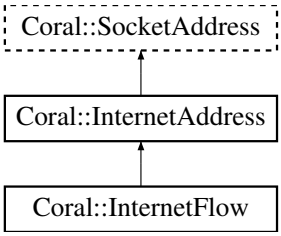
- internetaddress.h
- internetaddress.cc

6.31 Coral::InternetFlow Class Reference

Internet Flow.

`#include <internetflow.h>`

Inheritance diagram for Coral::InternetFlow:



Public Member Functions

- InternetFlow ()
- InternetFlow (const InternetFlow &flow)
- InternetFlow (const InternetAddress &address, const card32 flowLabel, const card8 trafficClass)
- void reset ()
- String getAddressString () 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

6.31.1 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

6.31.2 Constructor & Destructor Documentation

6.31.2.1 Coral::InternetFlow::InternetFlow ( )

Constructor for a new InternetFlow.

6.31.2.2 Coral::InternetFlow::InternetFlow (const InternetFlow & *flow*)

Constructor for a new InternetFlow.

Parameters

flow	InternetFlow to be copied.

6.31.2.3 Coral::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).

6.31.3 Member Function Documentation

6.31.3.1 String Coral::InternetFlow::getAddressString ( ) const [virtual]

Get address string.

Returns

Address string.

Reimplemented from Coral::InternetAddress.

6.31.3.2 card32 Coral::Coral::InternetFlow::getFlowInfo ( ) const [inline]

Get IPv6 flow info: (flowLabel | (trafficClass << 20)).

Returns

Flow info.

6.31.3.3 card32 Coral::Coral::InternetFlow::getFlowLabel ( ) const [inline]

Get flow label.

Returns

Flow label.

6.31.3.4 cardinal Coral::InternetFlow::getSystemAddress (sockaddr * *buffer*, const socklen_t *length*, const cardinal *type*) const [virtual]

getSystemAddress() implementation of SocketAddressInterface.

See also

SocketAddressInterface::getSystemAddress

Reimplemented from Coral::InternetAddress.

6.31.3.5 card8 Coral::Coral::InternetFlow::getTrafficClass ( ) const [inline]

Get traffic class.

Returns

Traffic class.

6.31.3.6 void Coral::InternetFlow::reset ( ) [virtual]

Reset flow info.

Reimplemented from Coral::InternetAddress.

6.31.3.7 void Coral::Coral::InternetFlow::setFlowLabel (const card32 *flowLabel*) [inline]

Set flow label.

Parameters

flowLabel	Flow label.

6.31.3.8 bool Coral::InternetFlow::setSystemAddress (sockaddr * *address*, socklen_t *length*) [virtual]

setSystemAddress() implementation of SocketAddressInterface.

See also

SocketAddressInterface::setSystemAddress

Reimplemented from Coral::InternetAddress.

6.31.3.9 void Coral::Coral::InternetFlow::setTrafficClass (const card8 *trafficClass*) [inline]

Set traffic class.

Parameters

trafficClass	New traffic class.

6.31.4 Member Data Documentation

6.31.4.1 card32 Coral::InternetFlow::FlowInfo [private]

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

- internetflow.h
- internetflow.cc

6.32 Coral::IntervalHeader Struct Reference

Interval Header.

`#include <tdtf.h>`

Public Member Functions

- cardinal getOffsetLH (const cardinal layer) const
- cardinal getOffsetEEBR (const cardinal layer) const
- cardinal getOffsetEEFC (const cardinal layer) const

Static Public Member Functions

- static cardinal getIntervalHeaderSize (const cardinal layers)

Public Attributes

- card32 Position
- card32 Length
- card8 Layers
- card8 Flags
- card16 pad
- card32 IntervalDescriptionSize
- card32 Offset [0]

Static Public Attributes

- static const cardinal offsetUH = 0

6.32.1 Detailed Description

Interval Header.

This is the header for an interval description.

Author

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

6.32.2 Member Function Documentation

6.32.2.1 static cardinal Coral::IntervalHeader::getIntervalHeaderSize (const cardinal *layers*) [inline, static]

Calculate interval header size.

Parameters

layers	Layer count.

Returns

Interval header size.

6.32.2.2 cardinal Coral::IntervalHeader::getOffsetEEBR (const cardinal *layer*) const [inline]

Get byterate empirical envelope offset number.

Parameters

layer	Layer.

Returns

Byterate empirical envelope offset number.

6.32.2.3 cardinal Coral::IntervalHeader::getOffsetEEFC (const cardinal *layer*) const [inline]

Get frame count empirical envelope offset number.

Parameters

layer	Layer.

Returns

Frame count empirical envelope offset number.

6.32.2.4 cardinal Coral::IntervalHeader::getOffsetLH (const cardinal *layer*) const [inline]

Get layer header offset number.

Parameters

layer	Layer.

Returns

Layer offset number.

6.32.3 Member Data Documentation

6.32.3.1 card8 Coral::IntervalHeader::Flags

Interval flags.

6.32.3.2 card32 Coral::IntervalHeader::IntervalDescriptionSize

Total size of the interval description.

6.32.3.3 card8 Coral::IntervalHeader::Layers

Number of layers.

6.32.3.4 card32 Coral::IntervalHeader::Length

Interval length.

6.32.3.5 card32 Coral::IntervalHeader::Offset[0]

Offset to interval descriptions: 0: Utilization header for frame rate utilization 1 1 + (0 ∗ Layers): Layer header für Layer #0 2 2 + (0 ∗ Layers): Empirical Envelope header für Layer #0 3 3 + (0 ∗ Layers): Frame count Empirical Envelope header for layer #0 4: 1 + (1 ∗ Layers): Layer header für Layer #1 5: 2 + (1 ∗ Layers): Empirical Envelope header

für Layer #1 6: 3 + (1 ∗ Layers): Frame count Empirical Envelope header for layer #1 ...

6.32.3.6 const cardinal Coral::IntervalHeader::offsetUH = 0 [static]

Utilization header offset number.

6.32.3.7 card16 Coral::IntervalHeader::pad

Unused. Should be set to 0.

6.32.3.8 card32 Coral::IntervalHeader::Position

Interval start position.

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

- tdtf.h

6.33 Coral::LayerClassMapping Struct Reference

Layer Class Mapping.

`#include <resourceutilizationpoint.h>`

Public Attributes

- cardinal Possibilities
- LayerClassMappingPossibility Possibility [TrafficClassValues::MaxValues]

6.33.1 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

6.33.2 Member Data Documentation

6.33.2.1 cardinal Coral::LayerClassMapping::Possibilities

Number of possible DiffServ classes for the layer.

6.33.2.2 LayerClassMappingPossibility Coral::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

6.34 Coral::LayerClassMappingPossibility Struct Reference

Layer Class Mapping Possibility.

`#include <resourceutilizationpoint.h>`

Public Attributes

- cardinal Class
- cardinal BufferDelay
- double Cost
- card64 Bandwidth

6.34.1 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

6.34.2 Member Data Documentation

6.34.2.1 card64 Coral::LayerClassMappingPossibility::Bandwidth

Bandwidth.

6.34.2.2 cardinal Coral::LayerClassMappingPossibility::BufferDelay

Buffer delay.

6.34.2.3 cardinal Coral::LayerClassMappingPossibility::Class

Class ID.

6.34.2.4 double Coral::LayerClassMappingPossibility::Cost

Cost.

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

- resourceutilizationpoint.h

6.35 Coral::LayerHeader Struct Reference

Layer Header.

`#include <tdtf.h>`

Public Attributes

- card32 FrameSizeUtilizationOffset
- card32 Scalability
- card8 Flags
- card8 pad

6.35.1 Detailed Description

Layer Header.

This is the header for a layer description.

Author

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

6.35.2 Member Data Documentation

6.35.2.1 card8 Coral::LayerHeader::Flags

Layer flags.

6.35.2.2 card32 Coral::LayerHeader::FrameSizeUtilizationOffset

Offset of frame size utilization header.

6.35.2.3 card8 Coral::LayerHeader::pad

Reserved byte.

6.35.2.4 card32 Coral::LayerHeader::Scalability

Scalability (out of (0,1]) divided by 2^30. Minimum FrameSize = Scalability * FrameSize.

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

- tdtf.h

6.36 Coral::MainIndexEntry Struct Reference

Layer Header.

`#include <tdtf.h>`

Public Attributes

- card64 FrameRate
- card32 Trace
- card32 Intervals

6.36.1 Detailed Description

Layer Header.

This is an entry of the frame rate index.

Author

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

6.36.2 Member Data Documentation

6.36.2.1 card64 Coral::MainIndexEntry::FrameRate

Frame rate.

6.36.2.2 card32 Coral::MainIndexEntry::Intervals

File position of interval header.

6.36.2.3 card32 Coral::MainIndexEntry::Trace

File position of trace.

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

- tdtf.h

6.37 Coral::MainIndexHeader Struct Reference

Main Index Header.

`#include <tdtf.h>`

Public Attributes

- card32 Entries
- MainIndexEntry Entry [0]

6.37.1 Detailed Description

Main Index Header.

This is the header of the frame rate index.

Author

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

6.37.2 Member Data Documentation

6.37.2.1 card32 Coral::MainIndexHeader::Entries

Number of entries.

6.37.2.2 MainIndexEntry Coral::MainIndexHeader::Entry[0]

Array of main index entries.

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

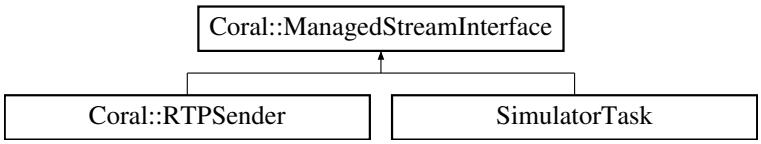
- tdtf.h

6.38 Coral::ManagedStreamInterface Class Reference

Managed Stream Interface.

`#include <managedstreaminterface.h>`

Inheritance diagram for Coral::ManagedStreamInterface:



Public Member Functions

- virtual AbstractQoSDescription * getQoSDescription (const card64 offset)=0
- virtual void updateQuality (const AbstractQoSDescription *aqd)=0
- virtual void lock ()=0
- virtual void unlock ()=0

6.38.1 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

6.38.2 Member Function Documentation

6.38.2.1 virtual AbstractQoSDescription* Coral::ManagedStreamInterface::getQoSDescription (const card64 *offset*) [pure virtual]

Get QoS description for current time + offset.

Parameters

offset	Offset in microseconds.

Returns

QoS Description.

Implemented in Coral::RTPSender, and SimulatorTask.

6.38.2.2 virtual void Coral::ManagedStreamInterface::lock ( ) [pure virtual]

Lock stream.

Implemented in Coral::RTPSender, and SimulatorTask.

6.38.2.3 virtual void Coral::ManagedStreamInterface::unlock ( ) [pure virtual]

Unlock stream.

Implemented in Coral::RTPSender, and SimulatorTask.

6.38.2.4 virtual void Coral::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 Coral::RTPSender, and SimulatorTask.

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

- managedstreaminterface.h

6.39 Coral::TDTFReader::MediaCacheEntry Struct Reference

`#include <tdtfreader.h>`

Public Attributes

- int InputFile
- char * InputMemory
- cardinal InputLength
- cardinal UserCount

6.39.1 Member Data Documentation

6.39.1.1 int Coral::TDTFReader::MediaCacheEntry::InputFile

6.39.1.2 cardinal Coral::TDTFReader::MediaCacheEntry::InputLength

6.39.1.3 char* Coral::TDTFReader::MediaCacheEntry::InputMemory

6.39.1.4 cardinal Coral::TDTFReader::MediaCacheEntry::UserCount

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

- tdtfreader.h

6.40 Coral::MediaInfo Class 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

6.40.1 Detailed Description

Media Info.

This class contains information on a media.

Author

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

6.40.2 Constructor & Destructor Documentation

6.40.2.1 Coral::MediaInfo::MediaInfo ( )

Constructor.

6.40.3 Member Function Documentation

6.40.3.1 void Coral::MediaInfo::reset ( )

Reset.

6.40.3.2 void Coral::MediaInfo::translate ( )

Translate byte order.

6.40.4 Member Data Documentation

6.40.4.1 char Coral::MediaInfo::Artist[MaxArtistLength+1]

Artist string.

6.40.4.2 char Coral::MediaInfo::Comment[MaxCommentLength+1]

Comment string.

6.40.4.3 card64 Coral::MediaInfo::EndTimeStamp

End time stamp of the media.

6.40.4.4 const cardinal Coral::MediaInfo::MaxArtistLength = 47 [static]

Constant for the maximum author length.

6.40.4.5 const cardinal Coral::MediaInfo::MaxCommentLength = 47 [static]

Constant for the maximum comment length.

6.40.4.6 const cardinal Coral::MediaInfo::MaxTitleLength = 47 [static]

Constant for the maximum title length.

6.40.4.7 card64 Coral::MediaInfo::StartTimeStamp

Start time stamp of the media.

6.40.4.8 char Coral::MediaInfo::Title[MaxTitleLength+1]

Title string.

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

- mediainfo.h
- mediainfo.cc

6.41 MediaList Struct Reference

Public Attributes

- String File
- String Name

6.41.1 Member Data Documentation

6.41.1.1 String MediaList::File

6.41.1.2 String MediaList::Name

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

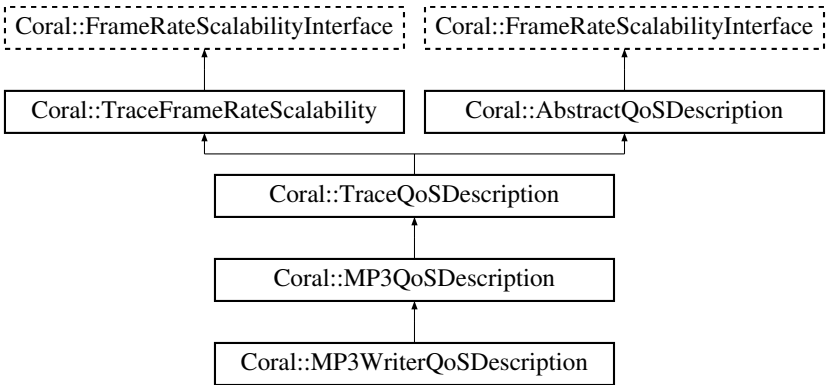
- delayanalyzer.cc
- differenceanalyzer.cc
- t7.cc

6.42 Coral::MP3QoSDescription Class Reference

MP3 QoS Description.

`#include <mp3qosdescription.h>`

Inheritance diagram for Coral::MP3QoSDescription:



Public Member Functions

- void calculateMaxUtilizationForBandwidthArray (const card64 *totalBandwidth-Array, ResourceUtilizationPoint *rupArray, const cardinal points) const

6.42.1 Detailed Description

MP3 QoS Description.

This class extends TraceWriterQoSDescription by MP3-specific utilization calculation necessary to generate resource/utilization lists.

Author

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

6.42.2 Member Function Documentation

6.42.2.1 void Coral::MP3QoSDescription::calculateMaxUtilizationForBandwidthArray (const card64 * *totalBandwidthArray,* ResourceUtilizationPoint * *rupArray,* const cardinal *points*) const [virtual]

Implementation of AbstractQoSDescription's calculateUtilizationForLayerBandwidths().

See also

AbstractQoSDescription::calculateUtilizationForLayerBandwidths

Reimplemented from Coral::AbstractQoSDescription.

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

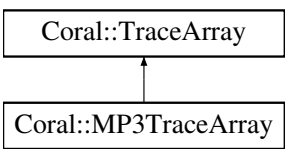
- mp3qosdescription.h
- mp3qosdescription.cc

6.43 Coral::MP3TraceArray Class Reference

MP3 Trace Array.

`#include <mp3tracearray.h>`

Inheritance diagram for Coral::MP3TraceArray:



Public Member Functions

- MP3TraceArray (const TraceConfiguration &config)
- bool load (const char *name)

6.43.1 Detailed Description

MP3 Trace Array.

This is an array of MP3 layer traces.

Author

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

6.43.2 Constructor & Destructor Documentation

6.43.2.1 Coral::MP3TraceArray::MP3TraceArray (const TraceConfiguration & *config*)

Constructor.

6.43.3 Member Function Documentation

6.43.3.1 bool Coral::MP3TraceArray::load (const char * *name*)

Load MP3 trace.

Parameters

name	File name.

Returns

true, if load has been successful; false otherwise.

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

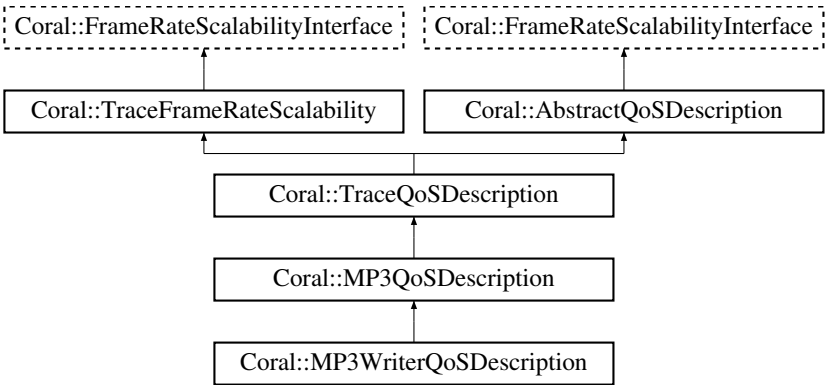
- mp3tracearray.h
- mp3tracearray.cc

6.44 Coral::MP3WriterQoSDescription Class Reference

MP3 Writer QoS Description.

`#include <mp3writerqosdescription.h>`

Inheritance diagram for Coral::MP3WriterQoSDescription:



Public Member Functions

- MP3WriterQoSDescription (const TraceConfiguration *traceConfiguration)
- double calculateUtilizationForLayerBandwidths (const double frameRate, const cardinal layers, const cardinal *bandwidth) const

Private Attributes

- const TraceConfiguration * Config

6.44.1 Detailed Description

MP3 Writer QoS Description.

This is the QoS description of a trace, extended by methods for MP3 resource/utilization calculation. These overwrite the ones which are used for calculating utilization from the trace file by the ones which use a new configuration. This is necessary to update the utilization of the file.

Author

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

6.44.2 Constructor & Destructor Documentation

6.44.2.1 Coral::MP3WriterQoSDescription::MP3WriterQoSDescription (const TraceConfiguration * *traceConfiguration*)

Constructor.

Parameters

trace-Configuration	Configuration.

6.44.3 Member Function Documentation

6.44.3.1 double Coral::MP3WriterQoSDescription::calculateUtilizationForLayer-Bandwidths (const double *frameRate*, const cardinal *layers*, const cardinal * *bandwidth*) const

Implementation of AbstractQoSDescription's calculateUtilizationForLayerBandwidths().

See also

AbstractQoSDescription::calculateUtilizationForLayerBandwidths

6.44.4 Member Data Documentation

6.44.4.1 const TraceConfiguration* Coral::MP3WriterQoSDescription::Config [private]

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

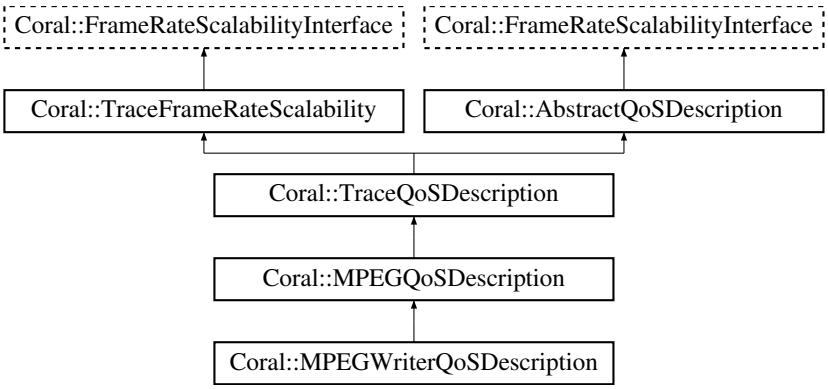
- mp3writerqosdescription.h
- mp3writerqosdescription.cc

6.45 Coral::MPEGQoSDescription Class Reference

MPEG QoS Description.

`#include <mpegqosdescription.h>`

Inheritance diagram for Coral::MPEGQoSDescription:



Public Member Functions

- void calculateMaxUtilizationForBandwidthArray (const card64 *totalBandwidthArray, ResourceUtilizationPoint *rupArray, const cardinal points) const

Private Member Functions

- bool tryAllocation (card64 &available, card64 *requiredArray, card64 *bandwidthArray, cardinal layers) const

6.45.1 Detailed Description

MPEG QoS Description.

This class extends TraceWriterQoSDescription by MPEG-specific utilization calculation necessary to generate resource/utilization lists.

Author

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

6.45.2 Member Function Documentation

6.45.2.1 void Coral::MPEGQoSDescription::calculateMaxUtilizationForBandwidthArray (const card64 * *totalBandwidthArray*, ResourceUtilizationPoint * *rupArray*, const cardinal *points*) const [virtual]

Implementation of AbstractQoSDescription's calculateUtilizationForLayerBandwidths().

See also

AbstractQoSDescription::calculateUtilizationForLayerBandwidths

Reimplemented from Coral::AbstractQoSDescription.

6.45.2.2 bool Coral::MPEGQoSDescription::tryAllocation (card64 & *available*, card64 * *requiredArray*, card64 * *bandwidthArray*, cardinal *layers*) const [private]

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

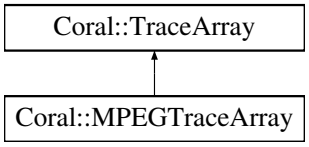
- mpegqosdescription.h
- mpegqosdescription.cc

6.46 Coral::MPEGTraceArray Class Reference

MPEG Trace Array.

`#include <mpegtracearray.h>`

Inheritance diagram for Coral::MPEGTraceArray:



Public Member Functions

- MPEGTraceArray (const TraceConfiguration &config)
- bool load (const char *name, const char *framePattern, const cardinal frameRate, const cardinal extLayers, const double fakeE1, const double fakeE2)
- TraceArray * MPEGTraceArray::decreaseFrameRate () const

Static Public Attributes

- static const cardinal LayerMPEGBaseI = 0
- static const cardinal LayerMPEGBaseP = 1
- static const cardinal LayerMPEGBaseB = 2
- static const cardinal LayerMPEGExtI1 = 3
- static const cardinal LayerMPEGExtP1 = 4
- static const cardinal LayerMPEGExtB1 = 5
- static const cardinal LayerMPEGExtI2 = 6
- static const cardinal LayerMPEGExtP2 = 7
- static const cardinal LayerMPEGExtB2 = 8

6.46.1 Detailed Description

MPEG Trace Array.

This is an array of MPEG layer traces.

Author

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

6.46.2 Constructor & Destructor Documentation

6.46.2.1 Coral::MPEGTraceArray::MPEGTraceArray (const TraceConfiguration & *config*)

Constructor.

6.46.3 Member Function Documentation

6.46.3.1 bool Coral::MPEGTraceArray::load (const char * *name*, const char * *framePattern*, const cardinal *frameRate*, const cardinal *extLayers*, const double *fakeE1*, const double *fakeE2*)

Load MPEG trace.

Parameters

name	File name.
framePattern	Frame pattern of the trace file.
frameRate	Frame rate of the trace file.
extLayers	Number of extension layers (0, 1 or 2).
fakeE1	Fake 1st extension layer: FrameSizeE1 = fakeE1 * FrameSizeBase.
fakeE2	Fake 2nd extension layer: FrameSizeE2 = fakeE2 * FrameSizeBase.

Returns

true, if load has been successful; false otherwise.

6.46.3.2 TraceArray* Coral::MPEGTraceArray::MPEGTraceArray::decreaseFrameRate ( ) const

Reimplementation of TraceArray's decreaseFrameRate() method.

See also

TraceArray::decreaseFrameRate

6.46.4 Member Data Documentation

6.46.4.1 const cardinal Coral::MPEGTraceArray::LayerMPEGBaseB = 2 [static]

MPEG-I/II B-frames layer.

6.46.4.2 const cardinal Coral::MPEGTraceArray::LayerMPEGBaseI = 0 [static]

MPEG-I/II I-frames layer.

6.46.4.3 const cardinal Coral::MPEGTraceArray::LayerMPEGBaseP = 1 [static]

MPEG-I/II P-frames layer.

6.46.4.4 const cardinal Coral::MPEGTraceArray::LayerMPEGExtB1 = 5 [static]

MPEG-II B-frames 1st extension layer.

6.46.4.5 const cardinal Coral::MPEGTraceArray::LayerMPEGExtB2 = 8 [static]

MPEG-II B-frames 2nd extension layer.

6.46.4.6 const cardinal Coral::MPEGTraceArray::LayerMPEGExtI1 = 3 [static]

MPEG-II I-frames 1st extension layer.

6.46.4.7 const cardinal Coral::MPEGTraceArray::LayerMPEGExtI2 = 6 [static]

MPEG-II I-frames 2nd extension layer.

6.46.4.8 const cardinal Coral::MPEGTraceArray::LayerMPEGExtP1 = 4 [static]

MPEG-II P-frames 1st extension layer.

6.46.4.9 const cardinal Coral::MPEGTraceArray::LayerMPEGExtP2 = 7 [static]

MPEG-II P-frames 2nd extension layer.

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

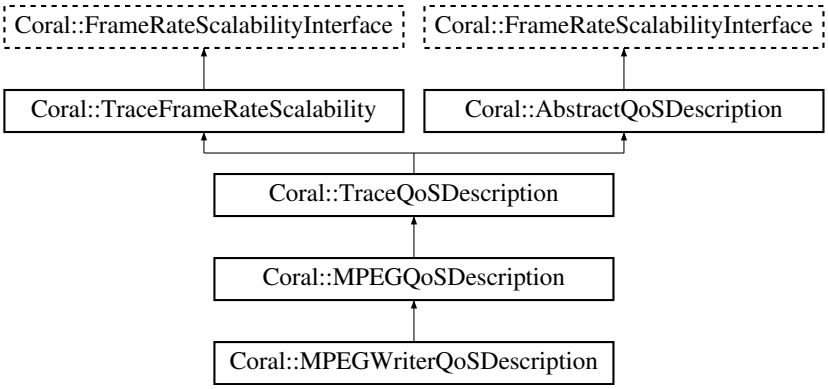
- mpegtracearray.h
- mpegtracearray.cc

6.47 Coral::MPEGWriterQoSDescription Class Reference

MPEG Writer QoS Description.

`#include <mpegwriterqosdescription.h>`

Inheritance diagram for Coral::MPEGWriterQoSDescription:



Public Member Functions

- MPEGWriterQoSDescription (const TraceConfiguration *traceConfiguration)
- double calculateUtilizationForLayerBandwidths (const double frameRate, const cardinal layers, const cardinal *bandwidth) const

Private Attributes

- const TraceConfiguration * Config

6.47.1 Detailed Description

MPEG Writer QoS Description.

This is the QoS description of a trace, extended by methods for MPEG resource/utilization calculation. These overwrite the ones which are used for calculating utilization from

the trace file by the ones which use a new configuration. This is necessary to update the utilization of the file.

Author

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

6.47.2 Constructor & Destructor Documentation

6.47.2.1 Coral::MPEGWriterQoSDescription::MPEGWriterQoSDescription (const TraceConfiguration * traceConfiguration)

Constructor.

Parameters

traceConfiguration	Configuration.

6.47.3 Member Function Documentation

6.47.3.1 double Coral::MPEGWriterQoSDescription::calculateUtilizationForLayerBandwidths (const double *frameRate*, const cardinal *layers*, const cardinal * *bandwidth*) const

Implementation of AbstractQoSDescription's calculateUtilizationForLayerBandwidths().

See also

AbstractQoSDescription::calculateUtilizationForLayerBandwidths

6.47.4 Member Data Documentation

6.47.4.1 const TraceConfiguration* Coral::MPEGWriterQoSDescription::Config [private]

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

- mpegwriterqosdescription.h
- mpegwriterqosdescription.cc

6.48 Coral::RoundTripTimePinger::Ping4Packet Struct Reference

Public Attributes

- icmphdr Header
- card64 TimeStamp

6.48.1 Member Data Documentation

6.48.1.1 icmphdr Coral::RoundTripTimePinger::Ping4Packet::Header

6.48.1.2 card64 Coral::RoundTripTimePinger::Ping4Packet::TimeStamp

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

- roundtriptimepinger.h

6.49 Coral::RoundTripTimePinger::Ping6Packet Struct Reference

Public Attributes

- icmp6_hdr Header
- card64 TimeStamp

6.49.1 Member Data Documentation

6.49.1.1 icmp6_hdr Coral::RoundTripTimePinger::Ping6Packet::Header

6.49.1.2 card64 Coral::RoundTripTimePinger::Ping6Packet::TimeStamp

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

- roundtriptimepinger.h

6.50 Coral::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

6.50.1 Detailed Description

PingerHost.

This structure contains internal information for RoundTripTimePinger.

Author

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

6.50.2 Member Data Documentation

6.50.2.1 InternetAddress Coral::PingerHost::Address

InternetAddress to send ping to.

6.50.2.2 String Coral::PingerHost::AddressString

The ping address in text format.

6.50.2.3 bool Coral::PingerHost::IsIPv6

Does this address use IPv6?

6.50.2.4 card64 Coral::PingerHost::LastEchoTimeStamp

Timestamp of last received ping.

6.50.2.5 card64 Coral::PingerHost::LastPingTimeStamp

Timestamp of last sent ping.

6.50.2.6 cardinal Coral::PingerHost::MaxRawRoundTripTime

Maximum raw round trip time (directly calculated from packet).

6.50.2.7 cardinal Coral::PingerHost::RoundTripTime

Round trip time.

6.50.2.8 card16 Coral::PingerHost::SeqNum

Sequence number.

6.50.2.9 card8 Coral::PingerHost::TrafficClass

Traffic class.

6.50.2.10 cardinal Coral::PingerHost::UserCount

User counter (number of addHost() calls for this destination).

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

- pingerhost.h

6.51 Coral::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

6.51.1 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

6.51.2 Member Function Documentation

6.51.2.1 int Coral::PortableAddress::operator!= (const PortableAddress & *address*) const

Implementation of == operator.

6.51.2.2 int Coral::PortableAddress::operator< (const PortableAddress & *address*) const

Implementation of < operator.

6.51.2.3 int Coral::PortableAddress::operator<= (const PortableAddress & *address*) const

Implementation of <= operator.

6.51.2.4 int Coral::PortableAddress::operator== (const PortableAddress & *address*) const

Implementation of == operator.

6.51.2.5 int Coral::PortableAddress::operator> (const PortableAddress & *address*) const

Implementation of > operator.

6.51.2.6 int Coral::PortableAddress::operator>= (const PortableAddress & *address*) const

Implementation of >= operator.

6.51.2.7 void Coral::PortableAddress::reset ( ) [inline]

Reset portable address.

6.51.3 Member Data Documentation

6.51.3.1 card16 Coral::PortableAddress::Host[8]

Host address in network byte order. IPv4 addresses are converted to IPv4-mapped IPv6 addresses.

6.51.3.2 card16 Coral::PortableAddress::Port

Port number.

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

- portableaddress.h

6.52 Coral::PositionLengthIntervalIndexEntry Struct Reference

Layer Header.

```
#include <tdtf.h>
```

Public Attributes

- card32 Position
- card32 Length
- card32 Interval

6.52.1 Detailed Description

Layer Header.

This is an entry of the Position/Length/Interval index.

Author

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

6.52.2 Member Data Documentation

6.52.2.1 card32 Coral::PositionLengthIntervalIndexEntry::Interval

File position of interval header.

6.52.2.2 card32 Coral::PositionLengthIntervalIndexEntry::Length

Interval length.

6.52.2.3 card32 Coral::PositionLengthIntervalIndexEntry::Position

Interval start position.

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

- tdtf.h

6.53 Coral::PositionLengthIntervalIndexHeader Struct Reference

Layer Header.

`#include <tdtf.h>`

Public Attributes

- card32 Entries
- PositionLengthIntervalIndexEntry Entry [0]

6.53.1 Detailed Description

Layer Header.

This is the header of the Position/Length/Interval index.

Author

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

6.53.2 Member Data Documentation

6.53.2.1 card32 Coral::PositionLengthIntervalIndexHeader::Entries

Number of entries.

6.53.2.2 PositionLengthIntervalIndexEntry Coral::PositionLengthIntervalIndexHeader::Entry[0]

Array of index entries.

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

- tdtf.h

6.54 Coral::QualityScenario Struct Reference

Public Attributes

- card16 MediaType
- card16 MediaSubtype
- card32 Flags
- cardinal MaxLayers
- QualityScenarioEntry Entry [TraceQoSDescription::MaxLayers]

6.54.1 Member Data Documentation

6.54.1.1 QualityScenarioEntry Coral::QualityScenario::Entry[TraceQoSDescription::MaxLayers]

6.54.1.2 card32 Coral::QualityScenario::Flags

6.54.1.3 cardinal Coral::QualityScenario::MaxLayers

6.54.1.4 card16 Coral::QualityScenario::MediaSubtype

6.54.1.5 card16 Coral::QualityScenario::MediaType

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

- traceencoder.cc

6.55 Coral::QualityScenarioEntry Struct Reference

Public Attributes

- double MaxLossRate
- double MaxJitter

6.55.1 Member Data Documentation

6.55.1.1 double Coral::QualityScenarioEntry::MaxJitter

6.55.1.2 double Coral::QualityScenarioEntry::MaxLossRate

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

- traceencoder.cc

6.56 Coral::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

6.56.1 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

6.56.2 Constructor & Destructor Documentation

6.56.2.1 Coral::Randomizer::Randomizer ( )

Constructor. Seed will be initialized by system timer (microseconds since January 01, 1970).

6.56.3 Member Function Documentation

6.56.3.1 double Coral::Randomizer::random ( ) [inline]

Get double random number out of interval [0,1].

Returns

The generated number.

6.56.3.2 cardinal Coral::Randomizer::random (const cardinal *a,* const cardinal *b*)

Get double random cardinal number out of interval [a,b].

Returns

The generated number.

6.56.3.3 double Coral::Randomizer::random (const double *a,* const double *b*)

Get double random double number out of interval [a,b].

Returns

The generated number.

6.56.3.4 card16 Coral::Randomizer::random16 ( ) [inline]

Get 16-bit random number.

Returns

The generated number.

6.56.3.5 card32 Coral::Randomizer::random32 ( ) [inline]

Get 32-bit random number.

Returns

The generated number.

6.56.3.6 card64 Coral::Randomizer::random64 ( ) [inline]

Get 64-bit random number.

Returns

The generated number.

6.56.3.7 card8 Coral::Randomizer::random8 ( ) [inline]

Get 8-bit random number.

Returns

The generated number.

6.56.3.8 void Coral::Randomizer::setSeed ( )

Set seed by system timer (microseconds since January 01, 1970).

6.56.3.9 void Coral::Randomizer::setSeed (const cardinal *seed*)

Set seed by given number.

Parameters

seed	Seed value.

6.56.4 Member Data Documentation

6.56.4.1 card32 Coral::Randomizer::Value [private]

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

- randomizer.h
- randomizer.cc

6.57 Range< T > Class Template Reference

Range.

`#include <range.h>`

Public Member Functions

- Range ()
- Range (const T min, const T max, const T value)
- void init (const T min, const T max, const T value)
- T getMin () const
- T getMax () const
- T getValue () const
- void setLimits (const T min, const T max)
- void setValue (const T value)
- Range< T > & operator= (const Range< T > &range)
- int operator== (const Range< T > &ti) const
- int operator!= (const Range< T > &ti) const

Public Attributes

- T Min
- T Max
- T Value

6.57.1 Detailed Description

template<class T>class Range< T >

Range.

This class implements the Range datatype template. It manages a value which has to be in the range from Min to Max. The only allowed exception is the value 0, which is available even if it is outside of the given range.

Author

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

6.57.2 Constructor & Destructor Documentation

6.57.2.1 template<class T > Range< T >::Range ( )

Default constructor.

6.57.2.2 template<class T> Range< T >::Range (const T *min*, const T *max*, const T *value*)

Create new range with given parameters.

Parameters

min	Minimum.
max	Maximum.
value	Value between Minimum and Maximum.

6.57.3 Member Function Documentation

6.57.3.1 template<class T> T Range< T >::getMax ( ) const [inline]

Get maximum.

Returns

Maximum.

6.57.3.2 template<class T> T Range< T >::getMin ( ) const [inline]

Get minimum.

Returns

Minimum.

6.57.3.3 template<class T> T Range< T >::getValue ( ) const [inline]

Get value.

Returns

Value.

6.57.3.4 template<class T> void Range< T >::init (const T *min*, const T *max*, const T *value*)

Initialize range with given parameters.

Parameters

min	Minimum.
max	Maximum.
value	Value between Minimum and Maximum.

6.57.3.5 template<class T> int Range< T >::operator!= (const Range< T > & *ti*) const [inline]

!= operator.

6.57.3.6 template<class T> Range<T>& Range< T >::operator= (const Range< T > & *range*)

Implementation of = operator

6.57.3.7 template<class T> int Range< T >::operator== (const Range< T > & *ti*) const [inline]

== operator.

6.57.3.8 template<class T> void Range< T >::setLimits (const T *min*, const T *max*) [inline]

Set limits.

Parameters

min	Minimum.
max	Maximum.

6.57.3.9 template<class T> void Range< T >::setValue (const T *value*) [inline]

Set value.

Parameters

value	Value.

6.57.4 Member Data Documentation

6.57.4.1 template<class T> T Range< T >::Max

6.57.4.2 template<class T> T Range< T >::Min

6.57.4.3 template<class T> T Range< T >::Value

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

- range.h
- range.cc

6.58 Coral::ResourceUtilizationEntry Struct Reference

Resource Utilization Entry.

`#include <tdtf.h>`

Public Attributes

- card32 Utilization
- card64 FrameRate
- card32 Bandwidth [0]

6.58.1 Detailed Description

Resource Utilization Entry.

This is an entry of a resource/utilization list.

Author

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

6.58.2 Member Data Documentation

6.58.2.1 card32 Coral::ResourceUtilizationEntry::Bandwidth[0]

Bandwidth (sum of all layers and for each layer).

6.58.2.2 card64 Coral::ResourceUtilizationEntry::FrameRate

Frame rate.

6.58.2.3 card32 Coral::ResourceUtilizationEntry::Utilization

Utilization divided by 2^30.

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

- tdtf.h

6.59 Coral::ResourceUtilizationHeader Struct Reference

Resource Utilization Header.

`#include <tdtf.h>`

Static Public Member Functions

- static cardinal getResourceUtilizationSize (const cardinal layers, const cardinal maxConstants)

Public Attributes

- card16 Entries
- card8 Layers
- card8 Flags
- card32 Position
- ResourceUtilizationEntry Entry [0]

6.59.1 Detailed Description

Resource Utilization Header.

This is the header for a resource/utilization list.

Author

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

6.59.2 Member Function Documentation

6.59.2.1 static cardinal Coral::ResourceUtilizationHeader::getResourceUtilizationSize (const cardinal *layers*, const cardinal *maxConstants*) [inline, static]

Calculate resource/utilization list size.

Parameters

layers	Number of layers.
maxPoints	Maximum number of RU points.

Returns

Resource/utilization list size.

6.59.3 Member Data Documentation

6.59.3.1 card16 Coral::ResourceUtilizationHeader::Entries

Number of entries.

6.59.3.2 ResourceUtilizationEntry Coral::ResourceUtilizationHeader::Entry[0]

Array of resource/utilization entires.

6.59.3.3 card8 Coral::ResourceUtilizationHeader::Flags

Flags.

6.59.3.4 card8 Coral::ResourceUtilizationHeader::Layers

Number of layers.

6.59.3.5 card32 Coral::ResourceUtilizationHeader::Position

Frame position of this header. If refers to the ∗maximum∗ frame rate!

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

- tdtf.h

6.60 Coral::ResourceUtilizationListIndexEntry Struct Reference

Resource Utilization List Index Entry.

`#include <tdtf.h>`

Public Attributes

- card32 Position
- card32 Length
- card32 List

6.60.1 Detailed Description

Resource Utilization List Index Entry.

This is an entry of the resource/utilization list index.

Author

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

6.60.2 Member Data Documentation

6.60.2.1 card32 Coral::ResourceUtilizationListIndexEntry::Length

Interval length in frame rate units.

6.60.2.2 card32 Coral::ResourceUtilizationListIndexEntry::List

File position of the resource/utilization list.

6.60.2.3 card32 Coral::ResourceUtilizationListIndexEntry::Position

Position (refering to ∗maximum∗ frame rate).

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

- tdtf.h

6.61 Coral::ResourceUtilizationListIndexHeader Struct Reference

Resource Utilization List Index Header.

`#include <tdtf.h>`

Public Attributes

- card32 Entries
- ResourceUtilizationListIndexEntry Entry [0]

6.61.1 Detailed Description

Resource Utilization List Index Header.

This is the resource/utilization list index header.

Author

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

6.61.2 Member Data Documentation

6.61.2.1 card32 Coral::ResourceUtilizationListIndexHeader::Entries

Number of entries.

6.61.2.2 ResourceUtilizationListIndexEntry Coral::ResourceUtilizationListIndexHeader::Entry[0]

Entries.

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

- tdtf.h

6.62 Coral::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

6.62.1 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

6.62.2 Member Function Documentation

6.62.2.1 int Coral::ResourceUtilizationMultiPoint::operator< (const ResourceUtilizationMultiPoint & *srup*) const [inline]

Operator "<".

6.62.2.2 int Coral::ResourceUtilizationMultiPoint::operator> (const ResourceUtilizationMultiPoint & *srup*) const [inline]

Operator ">".

6.62.3 Member Data Documentation

6.62.3.1 bool Coral::ResourceUtilizationMultiPoint::AlreadyAllocated

True, if this point has already been allocated during session's minimum bandwidth allocation.

6.62.3.2 card64 Coral::ResourceUtilizationMultiPoint::Bandwidth

Bandwidth.

6.62.3.3 double Coral::ResourceUtilizationMultiPoint::BandwidthCost

Bandwidth cost.

6.62.3.4 const cardinal Coral::ResourceUtilizationMultiPoint::MaxStreamsPerSession = 128 [static]

Maximum number of streams per session.

6.62.3.5 cardinal Coral::ResourceUtilizationMultiPoint::Point[MaxStreamsPerSession]

Array of point numbers for this multipoint's points.

6.62.3.6 SessionDescription* Coral::ResourceUtilizationMultiPoint::Session

SessionDescription of this multipoint's session.

6.62.3.7 double Coral::ResourceUtilizationMultiPoint::SessionPriorityFactor

Session's priority factor.

6.62.3.8 double Coral::ResourceUtilizationMultiPoint::SortingValue

Sorting value.

6.62.3.9 StreamDescription* Coral::ResourceUtilizationMultiPoint::Stream[MaxStreamsPerSession]

Array of StreamDescriptions for this multipoint's streams.

6.62.3.10 cardinal Coral::ResourceUtilizationMultiPoint::Streams

Number of streams in this session.

6.62.3.11 double Coral::ResourceUtilizationMultiPoint::Utilization

Utilization.

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

- bandwidthmanager.h

6.63 Coral::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 ResourceUtilizationPoint::sortResourceUtilizationList (ResourceUtilizationPoint *rup, const integer start, const integer end)
- static cardinal ResourceUtilizationPoint::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)

6.63.1 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

6.63.2 Member Function Documentation

6.63.2.1 static integer Coral::ResourceUtilizationPoint::ccw (const ResourceUtilizationPoint & *p0,* const ResourceUtilizationPoint & *p1,* const ResourceUtilizationPoint & *p2*) [inline, static, private]

6.63.2.2 cardinal Coral::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.

6.63.2.3 cardinal Coral::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.

6.63.2.4 int Coral::ResourceUtilizationPoint::operator!= (const ResourceUtilizationPoint & *rup*) const [inline]

Operator "!=".

6.63.2.5 int Coral::ResourceUtilizationPoint::operator== (const ResourceUtilizationPoint & *rup*) const [inline]

Operator "==".

6.63.2.6 void Coral::ResourceUtilizationPoint::reset ( )

Reset.

6.63.2.7 static cardinal Coral::ResourceUtilizationPoint::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.

6.63.2.8 static void Coral::ResourceUtilizationPoint::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.

6.63.2.9 static void Coral::ResourceUtilizationPoint::swapResourceUtilizationPoints (ResourceUtilizationPoint & *a,* ResourceUtilizationPoint & *b*) [inline, static, private]

6.63.3 Member Data Documentation

6.63.3.1 card64 Coral::ResourceUtilizationPoint::Bandwidth

Total bandwidth.

6.63.3.2 double Coral::ResourceUtilizationPoint::BandwidthCost

Bandwidth cost.

6.63.3.3 double Coral::ResourceUtilizationPoint::FrameRate

Frame rate.

6.63.3.4 BandwidthInfo Coral::ResourceUtilizationPoint::LayerBandwidthInfo[RTPConstants::RTPMaxQualityLayers]

Array of layers' bandwidth requirements.

6.63.3.5 cardinal Coral::ResourceUtilizationPoint::Layers

Number of layers.

6.63.3.6 LayerClassMapping Coral::ResourceUtilizationPoint::Mapping[RTP-Constants::RTPMaxQualityLayers]

Layer to DiffServ class mapping possibilities.

6.63.3.7 double Coral::ResourceUtilizationPoint::Utilization

Total utilization.

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

- resourceutilizationpoint.h
- resourceutilizationpoint.cc

6.64 Coral::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

6.64.1 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

6.64.2 Member Data Documentation

6.64.2.1 card64 Coral::ResourceUtilizationSimplePoint::Bandwidth

Bandwidth.

6.64.2.2 double Coral::ResourceUtilizationSimplePoint::BandwidthCost

Bandwidth cost.

6.64.2.3 cardinal Coral::ResourceUtilizationSimplePoint::Point

Point number (StreamDescription's RUList entry number).

6.64.2.4 double Coral::ResourceUtilizationSimplePoint::SortingValue

Sorting value.

6.64.2.5 StreamDescription* Coral::ResourceUtilizationSimplePoint::Stream

StreamDescription of this point's stream.

6.64.2.6 double Coral::ResourceUtilizationSimplePoint::StreamPriorityFactor

Stream's priority factor.

6.64.2.7 double Coral::ResourceUtilizationSimplePoint::Utilization

Utilization.

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

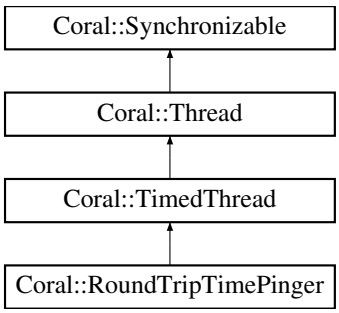
- bandwidthmanager.h

6.65 Coral::RoundTripTimePinger Class Reference

Round Trip Time Pinger.

`#include <roundtriptimepinger.h>`

Inheritance diagram for Coral::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 (ostream *scriptStream, ostream *dataStream, const char *dataName)
- void deactivateLogger ()
- bool isLogging () const
- void writeGPHeader (ostream &os, const char *dataName, const cardinal lineStyle=1)
- void writeGPData (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 RoundTripTimePinger::calculateChecksum (const card16 *addr, const cardinal length, card16 csum)
- card64 RoundTripTimePinger::sendPing4 (const InternetAddress &destination, const card8 trafficClass, const card16 sequenceNumber)
- card64 RoundTripTimePinger::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
- multiset< PingerHost > HostSet
- card64 GPHeaderTimeStamp
- bool Ready
- bool Logger
- ostream * LoggerScriptStream
- ostream * LoggerDataStream
- card64 MaxPingDelay
- Randomizer Random

Friends

- ostream & operator<< (ostream &os, RoundTripTimePinger &pinger)

6.65.1 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

6.65.2 Constructor & Destructor Documentation

6.65.2.1 Coral::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.

6.65.2.2 Coral::RoundTripTimePinger::~RoundTripTimePinger ( )

Destructor.

6.65.3 Member Function Documentation

6.65.3.1 void Coral::RoundTripTimePinger::activateLogger (ostream * *scriptStream,* 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).

6.65.3.2 bool Coral::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).

6.65.3.3 void Coral::RoundTripTimePinger::calculateRoundTripTime (const InternetAddress & *address*, const card8 *trafficClass*, const card64 *sendTime*, const card64 *arrivalTime*) [private]

6.65.3.4 void Coral::RoundTripTimePinger::checkUnreachable (PingerHost & *host*) [private]

6.65.3.5 void Coral::RoundTripTimePinger::deactivateLogger ( )

Deactivate logger.

6.65.3.6 double Coral::RoundTripTimePinger::getAlpha ( ) [inline]

Get constant alpha: RTT = alpha ∗ oldValue + (1 - alpha) ∗ newValue.

Returns

alpha.

6.65.3.7 cardinal Coral::RoundTripTimePinger::getHosts ( ) [inline]

Get number of hosts in RoundTripTimePinger.

Returns

Number of hosts.

6.65.3.8 card64 Coral::RoundTripTimePinger::getMaxPingDelay ( ) [inline]

Get maximum delay between two pings in microseconds.

Returns

Delay in microseconds.

6.65.3.9 cardinal Coral::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.

6.65.3.10 bool Coral::RoundTripTimePinger::isLogging ( ) const [inline]

Check, if logger is running.

6.65.3.11 bool Coral::RoundTripTimePinger::ready ( ) const [inline]

Check, if RoundTripTimePinger is ready.

Returns

true, if RoundTripTimePinger is ready; false otherwise.

6.65.3.12 bool Coral::RoundTripTimePinger::receiveEcho4 ( ) [private]

6.65.3.13 bool Coral::RoundTripTimePinger::receiveEcho6 ( ) [private]

6.65.3.14 void Coral::RoundTripTimePinger::removeHost (const InternetAddress & *address,* const card8 *trafficClass* = 0x00)

Remove host from RoundTripTimePinger list.

Parameters

address	Host address.
trafficClass	Traffic class.

6.65.3.15 card16 Coral::RoundTripTimePinger::RoundTripTimePinger::calculateChecksum (const card16 * *addr,* const cardinal *length,* card16 *csum*) [private]

6.65.3.16 card64 Coral::RoundTripTimePinger::RoundTripTimePinger::sendPing4 (const InternetAddress & *destination,* const card8 *trafficClass,* const card16 *sequenceNumber*) [private]

6.65.3.17 card64 Coral::RoundTripTimePinger::RoundTripTimePinger::sendPing6 (const InternetAddress & *destination,* const card8 *trafficClass,* const card16 *sequenceNumber*) [private]

6.65.3.18 void Coral::RoundTripTimePinger::setAlpha (const double *alpha*) [inline]

Set constant alpha: RTT = alpha * oldValue + (1 - alpha) * newValue.

Parameters

alpha	Alpha.

6.65.3.19 void Coral::RoundTripTimePinger::setMaxPingDelay (const card64 *delay*) [inline]

Set maximum delay between two pings in microseconds.

Parameters

delay	Delay in microseconds.

6.65.3.20 void Coral::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 Coral::TimedThread.

6.65.3.21 void Coral::RoundTripTimePinger::writeGPData (ostream & *os*)

Write GNUplot data line.

Parameters

os	Output stream.

6.65.3.22 void Coral::RoundTripTimePinger::writeGPHeader (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.

6.65.4 Friends And Related Function Documentation

6.65.4.1 ostream& operator<< (ostream & *os,* RoundTripTimePinger & *pinger*) [friend]

Friend output operator.

6.65.5 Member Data Documentation

6.65.5.1 card64 Coral::RoundTripTimePinger::GPHeaderTimeStamp [private]

6.65.5.2 multiset<PingerHost> Coral::RoundTripTimePinger::HostSet [private]

6.65.5.3 bool Coral::RoundTripTimePinger::Logger [private]

6.65.5.4 ostream* Coral::RoundTripTimePinger::LoggerDataStream [private]

6.65.5.5 ostream* Coral::RoundTripTimePinger::LoggerScriptStream [private]

6.65.5.6 card64 Coral::RoundTripTimePinger::MaxPingDelay [private]

6.65.5.7 const cardinal Coral::RoundTripTimePinger::MaxRoundTripTime = 180000000 [static]

Maximum round trip time in microseconds.

6.65.5.8 const card64 Coral::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).

6.65.5.9 Socket* Coral::RoundTripTimePinger::Ping4Socket [private]

6.65.5.10 Socket* Coral::RoundTripTimePinger::Ping6Socket [private]

6.65.5.11 Randomizer Coral::RoundTripTimePinger::Random [private]

6.65.5.12 bool Coral::RoundTripTimePinger::Ready [private]

6.65.5.13 double Coral::RoundTripTimePinger::RoundTripTimeAlpha [private]

6.65.5.14 const double Coral::RoundTripTimePinger::UnreachableFactor = 2.0 [static]

Unreachable Factor: 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).

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

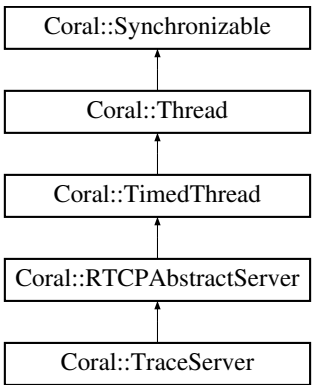
- roundtriptimepinger.h
- roundtriptimepinger.cc

6.66 Coral::RTCPAbstractServer Class Reference

RTCP abstract server.

`#include <rtcpabstractserver.h>`

Inheritance diagram for Coral::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 (const Client ∗client)=0
- virtual void appMessage (const Client ∗client, const char ∗name, const void ∗data, const cardinal dataLength)=0
- virtual void sdesMessage (const Client ∗client, const card8 type, const char ∗data, const cardinal length)=0
- virtual void receiverReport (const Client ∗client, const RTCPReceptionReport-Block ∗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, 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 Delete-Reason reason)
- Client ∗ findClient (const card32 source, const InternetFlow flow)
- void timerEvent ()

Private Attributes

- card64 DefaultTimeout
- multimap< const cardinal, Client ∗ > ClientSet

Friends

- class RTCPReceiver

6.66.1 Detailed Description

RTCP abstract server.

This class is an abstract RTCP server.

Author

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

6.66.2 Member Enumeration Documentation

6.66.2.1 enum Coral::RTCPAbstractServer::DeleteReason

A set of reasons for deleteClient() call.

Enumerator:

DeleteReason_UserBye

DeleteReason_Timeout

DeleteReason_Shutdown

DeleteReason_Error

6.66.3 Constructor & Destructor Documentation

6.66.3.1 Coral::RTCPAbstractServer::RTCPAbstractServer ( )

Constructor.

6.66.3.2 Coral::RTCPAbstractServer::~RTCPAbstractServer ( )

Destructor.

6.66.4 Member Function Documentation

6.66.4.1 virtual void Coral::RTCPAbstractServer::appMessage (const Client * *client*, const char * *name*, const 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 Coral::TraceServer.

6.66.4.2 virtual bool Coral::RTCPAbstractServer::checkClient (const 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 Coral::TraceServer.

6.66.4.3 virtual void Coral::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 Coral::TraceServer.

6.66.4.4 RTCPAbstractServer::Client * Coral::RTCPAbstractServer::findClient (const card32 *source,* const InternetFlow *flow*) [private]

6.66.4.5 card64 Coral::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.

6.66.4.6 cardinal Coral::RTCPAbstractServer::getMembers ( ) [inline]

Get number of members serverd by the server.

Returns

Number of members.

6.66.4.7 virtual void* Coral::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 Coral::TraceServer.

6.66.4.8 void Coral::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 Coral::TraceServer.

6.66.4.9 void Coral::RTCPAbstractServer::receivedApp (const InternetFlow *flow*, const card32 *source*, const char * *name*, const void * *data*, const card32 *dataLength*) [private]

6.66.4.10 void Coral::RTCPAbstractServer::receivedBye (const InternetFlow *flow*, const card32 *source*, const DeleteReason *reason*) [private]

6.66.4.11 void Coral::RTCPAbstractServer::receivedReceiverReport (const InternetFlow *flow*, const card32 *source*, const RTCPReceptionReportBlock * *report*, const cardinal *layer*) [private]

6.66.4.12 void Coral::RTCPAbstractServer::receivedSenderReport (const InternetFlow *flow*, const card32 *source*, const RTCPReceptionReportBlock * *report*, const cardinal *layer*) [private]

6.66.4.13 void Coral::RTCPAbstractServer::receivedSourceDescription (const InternetFlow *flow,* const card32 *source,* const card8 *type,* const char * *data,* const card8 *length*) [private]

6.66.4.14 virtual void Coral::RTCPAbstractServer::receiverReport (const Client * *client,* const 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 Coral::TraceServer.

6.66.4.15 virtual void Coral::RTCPAbstractServer::sdesMessage (const Client * *client,* const card8 *type,* const 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 Coral::TraceServer.

6.66.4.16 void Coral::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.

6.66.4.17 void Coral::RTCPAbstractServer::stop ( ) [virtual]

Reimplementation of Thread::stop() to remove all clients before stopping.

See also

Thread::stop

Reimplemented from Coral::TimedThread.

6.66.4.18 void Coral::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 Coral::TimedThread.

6.66.5 Friends And Related Function Documentation

6.66.5.1 friend class RTCPReceiver [friend]

RTCPReceiver is friend class to enable usage of receivedXXX() methods.

6.66.6 Member Data Documentation

6.66.6.1 multimap<const cardinal,Client*> Coral::RTCPAbstractServer::ClientSet [private]

6.66.6.2 card64 Coral::RTCPAbstractServer::DefaultTimeout [private]

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

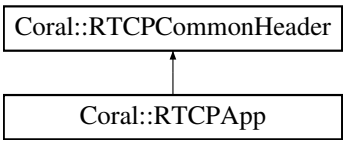
- rtcpabstractserver.h
- rtcpabstractserver.cc

6.67 Coral::RTCPApp Class Reference

RTCP APP Message.

`#include <rtcppacket.h>`

Inheritance diagram for Coral::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)

Private Attributes

- card32 Source
- char Name [4]
- char Data [0]

6.67.1 Detailed Description

RTCP APP Message.

This class manages an RTCP APP message

Author

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

See also

RTCPSender
RTCPReceiver
RTCPAbstractServer

6.67.2 Constructor & Destructor Documentation

6.67.2.1 Coral::RTCPApp::RTCPApp ( )

Constructor.

6.67.2.2 Coral::RTCPApp::RTCPApp (const card8 *subtype*)

Constructor.

Parameters

subtype	RTCP APP subtype.

6.67.3 Member Function Documentation

6.67.3.1 char* Coral::RTCPApp::getData ( ) [inline]

Get pointer to data field.

Returns

Pointer to data field.

6.67.3.2 char* Coral::RTCPApp::getName ( ) [inline]

Get pointer to name field.

Returns

Pointer to name field.

6.67.3.3 card32 Coral::RTCPApp::getSource ( ) const [inline]

Get source.

Returns

Source.

6.67.3.4 void Coral::RTCPApp::init (const card8 *subtype*)

6.67.3.5 void Coral::RTCPApp::setSource (const card32 *source*) [inline]

Set source.

Parameters

source	Source.

6.67.4 Member Data Documentation

6.67.4.1 char Coral::RTCPApp::Data[0] [private]

6.67.4.2 char Coral::RTCPApp::Name[4] [private]

6.67.4.3 card32 Coral::RTCPApp::Source [private]

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

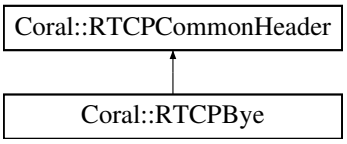
- rtcppacket.h
- rtcppacket.cc

6.68 Coral::RTCPBye Class Reference

RTCP BYE Message.

`#include <rtcppacket.h>`

Inheritance diagram for Coral::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 [0]

6.68.1 Detailed Description

RTCP BYE Message.

This class manages an RTCP BYE message

Author

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

See also

RTCPSender
RTCPReceiver
RTCPAbstractServer

6.68.2 Constructor & Destructor Documentation

6.68.2.1 Coral::RTCPBye::RTCPBye ( )

Constructor.

6.68.2.2 Coral::RTCPBye::RTCPBye (const card8 *count*)

Constructor.

Parameters

Count	count.

6.68.3 Member Function Documentation

6.68.3.1 card32 Coral::RTCPBye::getSource (const cardinal *index*) const [inline]

Get source at given index.

Parameters

index	Index.

Returns

Source.

6.68.3.2 void Coral::RTCPBye::init (const card8 *count*)

Initialize.

Parameters

Count	count.

6.68.3.3 void Coral::RTCPBye::setSource (const cardinal *index*, const card32 *source*) [inline]

Set source at given index.

Parameters

index	Index.
source	Source.

6.68.4 Member Data Documentation

6.68.4.1 card32 Coral::RTCPBye::Source[0] [private]

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

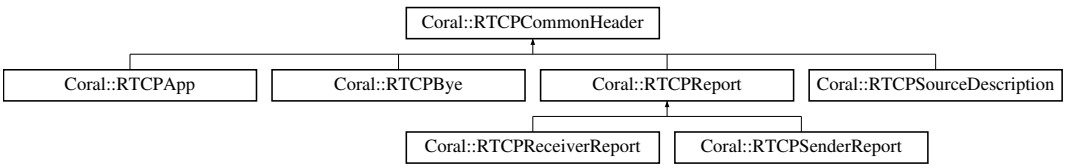
- rtcppacket.h
- rtcppacket.cc

6.69 Coral::RTCPCommonHeader Class Reference

RTCP Common Header.

`#include <rtcppacket.h>`

Inheritance diagram for Coral::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

6.69.1 Detailed Description

RTCP Common Header.

This class manages a common RTCP header.

Author

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

See also

RTCPSender
RTCPReceiver
RTCPAbstractServer

6.69.2 Constructor & Destructor Documentation

6.69.2.1 Coral::RTCPCommonHeader::RTCPCommonHeader ( )

Constructor.

6.69.3 Member Function Documentation

6.69.3.1 card8 Coral::RTCPCommonHeader::getCount ( ) const [inline]

Get count.

Returns

RTCP count.

6.69.3.2 card16 Coral::RTCPCommonHeader::getLength ( ) const [inline]

Get length.

Returns

RTCP Length.

6.69.3.3 card8 Coral::RTCPCommonHeader::getPacketType ( ) const [inline]

Get packet type.

Returns

RTCP packet type.

6.69.3.4 card8 Coral::RTCPCommonHeader::getPadding ( ) const [inline]

Get padding.

Returns

RTCP padding.

6.69.3.5 card8 Coral::RTCPCommonHeader::getVersion ( ) const [inline]

Get version.

Returns

RTCP version.

6.69.3.6 void Coral::RTCPCommonHeader::setCount (const card8 *count*) [inline]

Set count.

Parameters

count	RTCP count.

6.69.3.7 void Coral::RTCPCommonHeader::setLength (const card16 *length*) [inline]

Set length.

Parameters

length	RTCP Length.

6.69.3.8 void Coral::RTCPCommonHeader::setPacketType (const card8 *packetType*) [inline]

Set packetType.

Parameters

packetType	RTCP packet Type.

6.69.3.9 void Coral::RTCPCommonHeader::setPadding (const card8 *padding*) [inline]

Set padding.

Parameters

padding	RTCP padding.

6.69.3.10 void Coral::RTCPCommonHeader::setVersion (const card8 *version*) [inline]

Set version.

Parameters

version	RTCP version.

6.69.4 Member Data Documentation

6.69.4.1 card8 Coral::RTCPCommonHeader::C [protected]

6.69.4.2 card16 Coral::RTCPCommonHeader::Length [protected]

6.69.4.3 card8 Coral::RTCPCommonHeader::P [protected]

6.69.4.4 card8 Coral::RTCPCommonHeader::PT [protected]

6.69.4.5 card8 Coral::RTCPCommonHeader::V [protected]

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

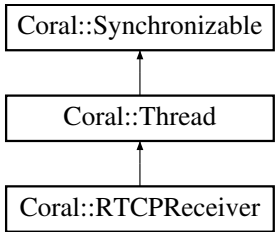
- rtcppacket.h
- rtcppacket.cc

6.70 Coral::RTCPReceiver Class Reference

RTCP Receiver.

`#include <rtcpreceiver.h>`

Inheritance diagram for Coral::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

6.70.1 Detailed Description

RTCP Receiver.

This class implements an RTCP receiver based on Thread.

Author

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

6.70.2 Constructor & Destructor Documentation

6.70.2.1 Coral::RTCPReceiver::RTCPReceiver ( )

Default constructor. You have to initialize RTCPReceiver by calling init(...) later!

See also

init

6.70.2.2 Coral::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.

6.70.2.3 Coral::RTCPReceiver::~RTCPReceiver ( )

Destructor.

6.70.3 Member Function Documentation

6.70.3.1 void Coral::RTCPReceiver::init (RTCPAbstractServer * *server*, Socket * *receiverSocket*)

Initialize 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.

6.70.3.2 void Coral::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 Coral::Thread.

6.70.4 Member Data Documentation

6.70.4.1 double Coral::RTCPReceiver::AverageRTCPSize [private]

6.70.4.2 Socket* Coral::RTCPReceiver::ReceiverSocket [private]

6.70.4.3 RTCPAbstractServer* Coral::RTCPReceiver::Server [private]

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

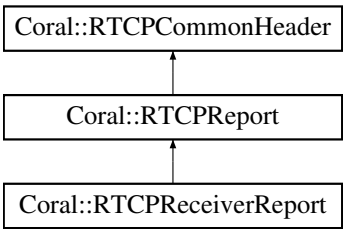
- rtcpreceiver.h
- rtcpreceiver.cc

6.71 Coral::RTCPReceiverReport Class Reference

RTCP Sender Report.

`#include <rtcppacket.h>`

Inheritance diagram for Coral::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 [0]

6.71.1 Detailed Description

RTCP Sender Report.

This class manages an RTCP receiver report

Author

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

See also

RTCPSender
RTCPReceiver
RTCPAbstractServer

6.71.2 Constructor & Destructor Documentation

6.71.2.1 Coral::RTCPReceiverReport::RTCPReceiverReport ( )

Constructor.

6.71.2.2 Coral::RTCPReceiverReport::RTCPReceiverReport (const card32 *syncSource*, const card8 *count* = 0)

Constructor.

Parameters

syncSource	SSRC.
count	Count.

6.71.3 Member Function Documentation

6.71.3.1 void Coral::RTCPReceiverReport::init (const card32 *syncSource*, const card8 *count* = 0)

Initialize.

Parameters

syncSource	SSRC.
count	Count.

6.71.4 Member Data Documentation

6.71.4.1 RTCPReceptionReportBlock Coral::RTCPReceiverReport::rr[0]

Array of RTCPReceptionReportBlocks

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

- rtcppacket.h
- rtcppacket.cc

6.72 Coral::RTCPReceptionReportBlock Class 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

6.72.1 Detailed Description

RTCP Reception Report Block.

This class manages a reception report block

Author

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

See also

RTCPSender
RTCPReceiver
RTCPAbstractServer

6.72.2 Constructor & Destructor Documentation

6.72.2.1 Coral::RTCPReceptionReportBlock::RTCPReceptionReportBlock ( )

Constructor.

6.72.2.2 Coral::RTCPReceptionReportBlock::RTCPReceptionReportBlock (const card32 *ssrc*)

Constructor.

Parameters

ssrc	SSRC.

6.72.3 Member Function Documentation

6.72.3.1 card32 Coral::RTCPReceptionReportBlock::getDLSR ( ) const [inline]

Get DLSR.

Returns

DLSR.

6.72.3.2 double Coral::RTCPReceptionReportBlock::getFractionLost ( ) const [inline]

Get fraction lost.

Returns

Fraction lost.

6.72.3.3 card32 Coral::RTCPReceptionReportBlock::getJitter ( ) const [inline]

Get jitter.

Returns

Jitter.

6.72.3.4 card32 Coral::RTCPReceptionReportBlock::getLastSeqNum ( ) const [inline]

Get last sequence number.

Returns

Last sequence number.

6.72.3.5 card32 Coral::RTCPReceptionReportBlock::getLSR ( ) const [inline]

Get LSR.

Returns

LSR.

6.72.3.6 card32 Coral::RTCPReceptionReportBlock::getPacketsLost ( ) const [inline]

Get packets lost.

Returns

Packets lost.

6.72.3.7 card32 Coral::RTCPReceptionReportBlock::getSSRC ( ) const [inline]

Get SSRC.

Returns

SSRC.

6.72.3.8 void Coral::RTCPReceptionReportBlock::init (const card32 *ssrc*)

Initialize.

Parameters

ssrc	SSRC.

6.72.3.9 void Coral::RTCPReceptionReportBlock::setDLSR (const card32 *dlsr*) [inline]

Set DLSR.

Parameters

dlsr	DLSR.

6.72.3.10 void Coral::RTCPReceptionReportBlock::setFractionLost (const double *fraction*) [inline]

Set fraction lost.

Parameters

fraction	Fraction lost.

6.72.3.11 void Coral::RTCPReceptionReportBlock::setJitter (const card32 *jitter*) [inline]

Set jitter.

Returns

jitter Jitter.

6.72.3.12 void Coral::RTCPReceptionReportBlock::setLastSeqNum (const card32 *lastSeq*) [inline]

Set last sequence number.

Parameters

lastSeq	Last sequence number.

6.72.3.13 void Coral::RTCPReceptionReportBlock::setLSR (const card32 *lsr*) [inline]

Set LSR.

Parameters

lsr	LSR.

6.72.3.14 void Coral::RTCPReceptionReportBlock::setPacketsLost (const card32 *packetsLost*) [inline]

Set packets lost.

Parameters

packetsLost	Packets lost.

6.72.3.15 void Coral::RTCPReceptionReportBlock::setSSRC (card32 *ssrc*) [inline]

Set SSRC.

Parameters

ssrc	SSRC.

6.72.4 Member Data Documentation

6.72.4.1 card32 Coral::RTCPReceptionReportBlock::DLSR [protected]

6.72.4.2 card32 Coral::RTCPReceptionReportBlock::Fraction [protected]

6.72.4.3 card32 Coral::RTCPReceptionReportBlock::Jitter [protected]

6.72.4.4 card32 Coral::RTCPReceptionReportBlock::LastSeq [protected]

6.72.4.5 card32 Coral::RTCPReceptionReportBlock::Lost [protected]

6.72.4.6 card32 Coral::RTCPReceptionReportBlock::LSR [protected]

6.72.4.7 card32 Coral::RTCPReceptionReportBlock::SSRC [protected]

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

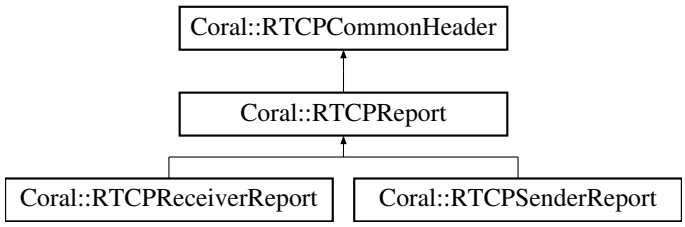
- rtcppacket.h
- rtcppacket.cc

6.73 Coral::RTCPReport Class Reference

RTCP Report.

`#include <rtcppacket.h>`

Inheritance diagram for Coral::RTCPReport:



Public Member Functions

- RTCPReport ()
- card32 getSSRC () const
- void setSSRC (const card32 ssrc)

Protected Attributes

- card32 SSRC

6.73.1 Detailed Description

RTCP Report.

This class manages an RTCP report.

Author

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

See also

RTCPSender
RTCPReceiver
RTCPAbstractServer

6.73.2 Constructor & Destructor Documentation

6.73.2.1 Coral::RTCPReport::RTCPReport ( )

Constructor.

6.73.3 Member Function Documentation

6.73.3.1 card32 Coral::RTCPReport::getSSRC ( ) const [inline]

Get SSRC.

Returns

SSRC.

6.73.3.2 void Coral::RTCPReport::setSSRC (const card32 *ssrc*) [inline]

Set SSRC.

Parameters

ssrc	SSRC.

6.73.4 Member Data Documentation

6.73.4.1 card32 Coral::RTCPReport::SSRC [protected]

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

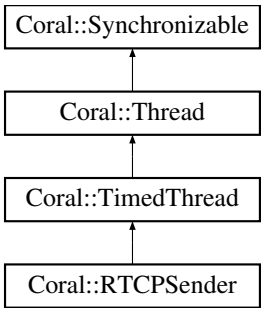
- rtcppacket.h
- rtcppacket.cc

6.74 Coral::RTCPSender Class Reference

RTCP Sender.

`#include <rtcpsender.h>`

Inheritance diagram for Coral::RTCPSender:



Public Member Functions

- RTCPSender ()
- RTCPSender (const card32 ssrc, Socket ∗senderSocket, RTPReceiver ∗receiver, const card64 bandwidth)
- ∼RTCPSender ()
- void init (const card32 ssrc, Socket ∗senderSocket, RTPReceiver ∗receiver, const card64 bandwidth)
- 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

- SocketAddress ∗ ReceiverAddress
- Socket ∗ SenderSocket
- RTPReceiver ∗ Receiver
- card32 SSRC
- multimap< const card8, RTCPSourceDescriptionItem ∗ > SDESItemSet

- Randomizer Random
- bool Initial
- bool WeSent
- integer Senders
- integer Members
- double RTCPBandwidth
- double AverageRTCPSize

6.74.1 Detailed Description

RTCP Sender.

This class implements an RTCP sender based on TimedThread.

Author

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

6.74.2 Constructor & Destructor Documentation

6.74.2.1 Coral::RTCPSender::RTCPSender ( )

Default constructor. You have to initialize RTPSender by calling init(...) later!

See also

init

6.74.2.2 Coral::RTCPSender::RTCPSender (const card32 *ssrc*, Socket * *senderSocket*, RTPReceiver * *receiver*, const card64 *bandwidth*)

Constructor for new RTCPSender. The new sender's thread has to be started by calling start()!

Parameters

ssrc	SSRC.
sender-Socket	Socket to write data to.
receiver	RTPReceiver for reports to send.
bandwidth	RTCP Bandwidth (see RFC 1889).

6.74.2.3 Coral::RTCPSender::~RTCPSender ( )

Destructor.

6.74.3 Member Function Documentation

6.74.3.1 bool Coral::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

6.74.3.2 double Coral::RTCPSender::computeTransmissionInterval ( ) [private]

6.74.3.3 void Coral::RTCPSender::init (const card32 *ssrc*, Socket * *senderSocket*, RTPReceiver * *receiver*, const card64 *bandwidth*)

Initialize new RTCPSender. The new sender's thread has to be started by calling start()!

Parameters

ssrc	SSRC.
senderSocket	Socket to write data to.
receiver	RTPReceiver for reports to send.
bandwidth	RTCP Bandwidth (see RFC 1889).

6.74.3.4 void Coral::RTCPSender::removeSDESItem (const card8 *type*)

Remove SDES item from SDES item list.

Parameters

type	SDES item type to be removed.

See also

addSDESItem
sendSDES

6.74.3.5 integer Coral::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.

6.74.3.6 integer Coral::RTCPSender::sendBye ( )

Send RTCP BYE message.

Returns

Bytes sent.

6.74.3.7 integer Coral::RTCPSender::sendReport ( )

Send RTCP receiver report from the SourceStateInfo given in the constructor.

Returns

Bytes sent.

6.74.3.8 integer Coral::RTCPSender::sendSDES ( )

Send RTCP SDES message from the list given by addSDESItem().

Returns

Bytes sent.

See also

addSDESItem

6.74.3.9 void Coral::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 Coral::TimedThread.

6.74.4 Member Data Documentation

6.74.4.1 double Coral::RTCPSender::AverageRTCPSize [private]

6.74.4.2 bool Coral::RTCPSender::Initial [private]

6.74.4.3 integer Coral::RTCPSender::Members [private]

6.74.4.4 Randomizer Coral::RTCPSender::Random [private]

6.74.4.5 RTPReceiver* Coral::RTCPSender::Receiver [private]

6.74.4.6 SocketAddress* Coral::RTCPSender::ReceiverAddress [private]

6.74.4.7 double Coral::RTCPSender::RTCPBandwidth [private]

6.74.4.8 multimap<const card8,RTCPSourceDescriptionItem*> Coral::RTCPSender::SDESItemSet [private]

6.74.4.9 integer Coral::RTCPSender::Senders [private]

6.74.4.10 Socket* Coral::RTCPSender::SenderSocket [private]

6.74.4.11 card32 Coral::RTCPSender::SSRC [private]

6.74.4.12 bool Coral::RTCPSender::WeSent [private]

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

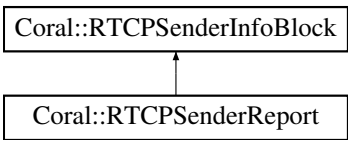
- rtcpsender.h
- rtcpsender.cc

6.75 Coral::RTCPSenderInfoBlock Class Reference

RTCP Sender Info Block.

`#include <rtcppacket.h>`

Inheritance diagram for Coral::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

6.75.1 Detailed Description

RTCP Sender Info Block.

This class manages a sender info block

Author

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

See also

RTCPSender
RTCPReceiver
RTCPAbstractServer

6.75.2 Constructor & Destructor Documentation

6.75.2.1 Coral::RTCPSenderInfoBlock::RTCPSenderInfoBlock ( )

Constructor.

6.75.3 Member Function Documentation

6.75.3.1 card64 Coral::RTCPSenderInfoBlock::getNTPTimeStamp ( ) const [inline]

Get NTP timestamp.

Returns

NTP timestamp.

6.75.3.2 card32 Coral::RTCPSenderInfoBlock::getOctetsSent ( ) const [inline]

Get octets sent.

Returns

Octets sent.

6.75.3.3 card32 Coral::RTCPSenderInfoBlock::getPacketsSent ( ) const [inline]

Get packets sent.

Returns

Packets sent.

6.75.3.4 card32 Coral::RTCPSenderInfoBlock::getRTPTimeStamp ( ) const [inline]

Get RTP time stamp.

Returns

RTP time stamp.

6.75.3.5 void Coral::RTCPSenderInfoBlock::setNTPTimeStamp (const card64 *timeStamp*) [inline]

Set NTP timestamp.

Parameters

timeStamp	NTP timestamp.

6.75.3.6 void Coral::RTCPSenderInfoBlock::setOctetsSent (const card32 *octets*) [inline]

Set octets sent.

Parameters

octets	Octets sent.

6.75.3.7 void Coral::RTCPSenderInfoBlock::setPacketsSent (const card32 *packets*) [inline]

Set packets sent.

Parameters

packets	Packets sent.

6.75.3.8 void Coral::RTCPSenderInfoBlock::setRTPTimeStamp (const card32 *timeStamp*) [inline]

Set RTP time stamp.

Parameters

timeStamp	RTP timestamp.

6.75.4 Member Data Documentation

6.75.4.1 card32 Coral::RTCPSenderInfoBlock::NTP_LeastSignificant [protected]

6.75.4.2 card32 Coral::RTCPSenderInfoBlock::NTP_MostSignificant [protected]

6.75.4.3 card32 Coral::RTCPSenderInfoBlock::OctetsSent [protected]

6.75.4.4 card32 Coral::RTCPSenderInfoBlock::PacketsSent [protected]

6.75.4.5 card32 Coral::RTCPSenderInfoBlock::RTPTimeStamp [protected]

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

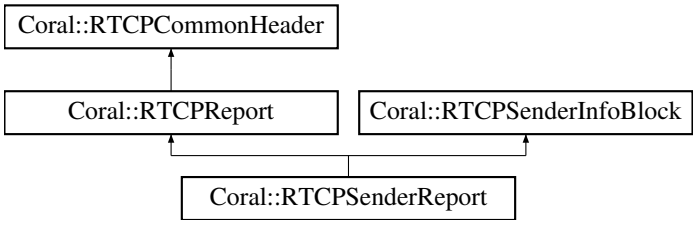
- rtcppacket.h
- rtcppacket.cc

6.76 Coral::RTCPSenderReport Class Reference

RTCP Sender Report.

`#include <rtcppacket.h>`

Inheritance diagram for Coral::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 [0]

6.76.1 Detailed Description

RTCP Sender Report.

This class manages an RTCP sender report

Author

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

See also

RTCPSender
RTCPReceiver
RTCPAbstractServer

6.76.2 Constructor & Destructor Documentation

6.76.2.1 Coral::RTCPSenderReport::RTCPSenderReport ( )

Constructor.

6.76.2.2 Coral::RTCPSenderReport::RTCPSenderReport (const card32 *syncSource*, const card8 *count* = 0)

Constructor.

Parameters

syncSource	SSRC.
count	Count.

6.76.3 Member Function Documentation

6.76.3.1 void Coral::RTCPSenderReport::init (const card32 *syncSource*, const card8 *count* = 0)

Initialize.

Parameters

syncSource	SSRC.
count	Count.

6.76.4 Member Data Documentation

6.76.4.1 RTCPReceptionReportBlock Coral::RTCPSenderReport::rr[0]

Array of RTCPReceptionReportBlocks

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

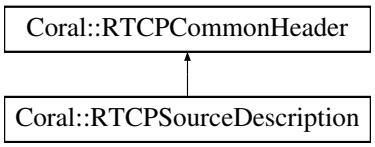
- rtcppacket.h
- rtcppacket.cc

6.77 Coral::RTCPSourceDescription Class Reference

RTCP Source Description (SDES)

`#include <rtcppacket.h>`

Inheritance diagram for Coral::RTCPSourceDescription:



Public Member Functions

- RTCPSourceDescription ()
- RTCPSourceDescription (const card8 count)
- void init (const card8 count)

Public Attributes

- RTCPSourceDescriptionChunk Chunk [1]

6.77.1 Detailed Description

RTCP Source Description (SDES)

This class manages an RTCP source description (SDES)

Author

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

See also

RTCPSender
RTCPReceiver
RTCPAbstractServer

6.77.2 Constructor & Destructor Documentation

6.77.2.1 Coral::RTCPSourceDescription::RTCPSourceDescription ( )

Constructor.

6.77.2.2 Coral::RTCPSourceDescription::RTCPSourceDescription (const card8 *count*)

Constructor.

Parameters

Count	count.

6.77.3 Member Function Documentation

6.77.3.1 void Coral::RTCPSourceDescription::init (const card8 *count*)

Initialize.

Parameters

Count	count.

6.77.4 Member Data Documentation

6.77.4.1 RTCPSourceDescriptionChunk Coral::RTCPSourceDescription::Chunk[1]

Array of SDES chunks.

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

- rtcppacket.h
- rtcppacket.cc

6.78 Coral::RTCPSourceDescriptionChunk Class Reference

RTCP Source Description Chunk.

`#include <rtcppacket.h>`

Public Attributes

- card32 SRC

- RTCPSourceDescriptionItem Item [1]

6.78.1 Detailed Description

RTCP Source Description Chunk.

This class manages an RTCP source description chunk

Author

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

See also

RTCPSender
RTCPReceiver
RTCPAbstractServer

6.78.2 Member Data Documentation

6.78.2.1 RTCPSourceDescriptionItem Coral::RTCPSourceDescriptionChunk::Item[1]

Array of SDES items.

6.78.2.2 card32 Coral::RTCPSourceDescriptionChunk::SRC

SSRC/CSRC Identifier of source.

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

- rtcppacket.h

6.79 Coral::RTCPSourceDescriptionItem Class Reference

RTCP Source Description Item.

`#include <rtcppacket.h>`

Public Attributes

- card8 Type
- card8 Length
- char Data [0]

6.79.1 Detailed Description

RTCP Source Description Item.

This class manages an RTCP source description item

Author

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

See also

RTCPSender
RTCPReceiver
RTCPAbstractServer

6.79.2 Member Data Documentation

6.79.2.1 char Coral::RTCPSourceDescriptionItem::Data[0]

Item data.

6.79.2.2 card8 Coral::RTCPSourceDescriptionItem::Length

Length in bytes.

6.79.2.3 card8 Coral::RTCPSourceDescriptionItem::Type

Item type (RTCP_SDES_...).

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

- rtcppacket.h

6.80 Coral::RTPPacket Class 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

- ostream & operator<< (ostream &os, const RTPPacket &packet)

6.80.1 Detailed Description

RTP Packet.

This class manages an RTP packet

Author

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

See also

RTPSender
RTPReceiver

6.80.2 Constructor & Destructor Documentation

6.80.2.1 Coral::RTPPacket::RTPPacket ( )

Constructor.

6.80.3 Member Function Documentation

6.80.3.1 cardinal Coral::RTPPacket::calculateHeaderSize ( ) const [inline]

Calculate header size.

Returns

Header size.

6.80.3.2 card32 Coral::RTPPacket::getCSRC (cardinal *index*) const [inline]

Get CSRC at given index.

Parameters

index	Index.

Returns

RTP CSRC.

6.80.3.3 card8 Coral::RTPPacket::getCSRCCount ( ) const [inline]

Get CSRC count.

Returns

RTP CSRC count.

6.80.3.4 card8 Coral::RTPPacket::getExtension ( ) const [inline]

Get extension.

Returns

RTP Extension.

6.80.3.5 bool Coral::RTPPacket::getMarker ( ) const [inline]

Get marker.

Returns

RTP Marker.

6.80.3.6 cardinal Coral::RTPPacket::getMaxPayloadSize ( ) const [inline]

Get maximum payload size.

Returns

Maximum payload size.

6.80.3.7 card8 Coral::RTPPacket::getPadding ( ) const [inline]

Get padding.

Returns

RTP Padding.

6.80.3.8 char* Coral::RTPPacket::getPayloadData ( ) const [inline]

Get pointer to payload data.

Returns

pointer to payload data.

6.80.3.9 card8 Coral::RTPPacket::getPayloadType ( ) const [inline]

Get payload type.

Returns

RTP Payload type.

6.80.3.10 card16 Coral::RTPPacket::getSequenceNumber ( ) const [inline]

Get sequence number.

Returns

RTP Sequence number.

6.80.3.11 card32 Coral::RTPPacket::getSSRC ( ) const [inline]

Get SSRC.

Returns

RTP SSRC.

6.80.3.12 card32 Coral::RTPPacket::getTimeStamp ( ) const [inline]

Get time stamp.

Returns

RTP Time stamp.

6.80.3.13 card8 Coral::RTPPacket::getVersion ( ) const [inline]

Get version.

Returns

RTP Version.

6.80.3.14 void Coral::RTPPacket::setCSRC (const cardinal *index*, const card32 *csrc*) [inline]

Set CSRC at given index.

Parameters

index	Index.
csrc	CSRC.

6.80.3.15 void Coral::RTPPacket::setCSRCCount (const card8 *count*) [inline]

Set CSRC count.

Parameters

count	RTP CSRC count.

6.80.3.16 void Coral::RTPPacket::setExtension (const card8 *extension*) [inline]

Set extension.

Parameters

extension	RTP Extension.

6.80.3.17 void Coral::RTPPacket::setMarker (const bool *marker*) [inline]

Set marker.

Parameters

marker	RTP Marker.

6.80.3.18 void Coral::RTPPacket::setPadding (const card8 *padding*) [inline]

Set padding.

Parameters

padding	RTP Padding.

6.80.3.19 void Coral::RTPPacket::setPayloadType (const card8 *payloadType*) [inline]

Set payload type.

Parameters

payloadType	RTP Payload type.

6.80.3.20 void Coral::RTPPacket::setSequenceNumber (const card16 *sequenceNumber*) [inline]

Set sequence number.

Parameters

sequence-Number	RTP Sequence number.

6.80.3.21 void Coral::RTPPacket::setSSRC (const card32 *ssrc*) [inline]

Set SSRC.

Parameters

ssrc	RTP SSRC.

6.80.3.22 void Coral::RTPPacket::setTimeStamp (const card32 *timeStamp*) [inline]

Set time stamp.

Parameters

timeStamp	RTP timeStamp.

6.80.3.23 void Coral::RTPPacket::setVersion (const card8 *version*) [inline]

Set version.

Parameters

version	RTP Version.

6.80.4 Friends And Related Function Documentation

6.80.4.1 ostream& operator<< (ostream & *os,* const RTPPacket & *packet*) [friend]

Output operator.

6.80.5 Member Data Documentation

6.80.5.1 card8 Coral::RTPPacket::CC [private]

6.80.5.2 card32 Coral::RTPPacket::CSRC[16] [private]

6.80.5.3 char Coral::RTPPacket::Data[RTPConstants::RTPMaxPayloadLimit] [private]

6.80.5.4 card8 Coral::RTPPacket::M [private]

6.80.5.5 card8 Coral::RTPPacket::P [private]

6.80.5.6 card8 Coral::RTPPacket::PT [private]

6.80.5.7 card16 Coral::RTPPacket::SequenceNumber [private]

6.80.5.8 card32 Coral::RTPPacket::SSRC [private]

6.80.5.9 card32 Coral::RTPPacket::TimeStamp [private]

6.80.5.10 card8 Coral::RTPPacket::V [private]

6.80.5.11 card8 Coral::RTPPacket::X [private]

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

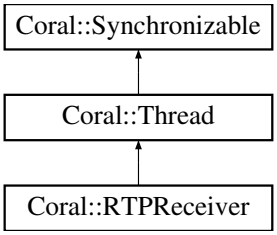
- rtppacket.h
- rtppacket.cc

6.81 Coral::RTPReceiver Class Reference

RTP Receiver.

`#include <rtpreceiver.h>`

Inheritance diagram for Coral::RTPReceiver:



Public Member Functions

- RTPReceiver ()
- RTPReceiver (DecoderInterface ∗decoder, Socket ∗receiverSocket)
- ∼RTPReceiver ()
- void init (DecoderInterface ∗decoder, Socket ∗receiverSocket)
- card64 getPosition () const
- card64 getMaxPosition () const
- card64 getBytesReceived (const cardinal layer) const
- card64 getPacketsReceived (const cardinal layer) const
- void resetBytesReceived (const cardinal layer)
- void resetPacketsReceived (const cardinal layer)
- cardinal getLayers () const
- InternetFlow getInternetFlow (const cardinal layer=0) const
- SourceStateInfo getSSI (const cardinal layer=0) const

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

6.81.1 Detailed Description

RTP Receiver.

This class implements an RTP receiver based on Thread.

Author

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

6.81.2 Constructor & Destructor Documentation

6.81.2.1 Coral::RTPReceiver::RTPReceiver ( )

Default constructor. You have to initialize RTPReceiver by calling init(...) later!

See also

init

6.81.2.2 Coral::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.
receiverSocket	Socket to receive data from.

See also

Thread::start

6.81.2.3 Coral::RTPReceiver::~RTPReceiver ( )

Destructor.

6.81.3 Member Function Documentation

6.81.3.1 card64 Coral::RTPReceiver::getBytesReceived (const cardinal *layer*) const [inline]

Get number of bytes received.

Parameters

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

Returns

Bytes received.

6.81.3.2 InternetFlow Coral::RTPReceiver::getInternetFlow (const cardinal *layer* = 0) const [inline]

Get InternetFlow of last transmission in a given layer.

Parameters

layer	Layer number.

Returns

InternetFlow.

6.81.3.3 cardinal Coral::RTPReceiver::getLayers ( ) const [inline]

Get number of layers of last transmission.

Returns

Number of layers.

6.81.3.4 card64 Coral::RTPReceiver::getMaxPosition ( ) const [inline]

Get maximum position of the encoder. The access is synchronized with the receiver thread.

Returns

Maximum position in nanoseconds.

6.81.3.5 card64 Coral::RTPReceiver::getPacketsReceived (const cardinal *layer*) const [inline]

Get number of packets received.

Parameters

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

Returns

Packets received.

6.81.3.6 card64 Coral::RTPReceiver::getPosition ( ) const [inline]

Get position of the encoder. The access is synchronized with the receiver thread.

Returns

Position in nanoseconds.

6.81.3.7 SourceStateInfo Coral::RTPReceiver::getSSI (const cardinal *layer* = 0) const [inline]

Get SourceStateInfo for given layer.

6.81.3.8 void Coral::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.
receiverSocket	Socket to receive data from.

See also

Thread::start

6.81.3.9 void Coral::RTPReceiver::resetBytesReceived (const cardinal *layer*) [inline]

Reset number of bytes received.

Parameters

layer	Layer number.

6.81.3.10 void Coral::RTPReceiver::resetPacketsReceived (const cardinal *layer*) [inline]

Reset number of packets received.

Parameters

layer	Layer number.

6.81.3.11 void Coral::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 Coral::Thread.

6.81.4 Friends And Related Function Documentation

6.81.4.1 friend class RTCPSender [friend]

RTCPSender is a friend class to enable efficient update of SSI data.

6.81.5 Member Data Documentation

6.81.5.1 card64 Coral::RTPReceiver::BytesReceived[RTPConstants::RTPMaxQualityLayers] [protected]

6.81.5.2 DecoderInterface* Coral::RTPReceiver::Decoder [private]

6.81.5.3 InternetFlow Coral::RTPReceiver::Flow[RTPConstants::RTPMaxQualityLayers] [protected]

6.81.5.4 cardinal Coral::RTPReceiver::Layers [protected]

6.81.5.5 card64 Coral::RTPReceiver::PacketsReceived[RTPConstants::RTPMaxQualityLayers] [protected]

6.81.5.6 Socket* Coral::RTPReceiver::ReceiverSocket [private]

6.81.5.7 SourceStateInfo Coral::RTPReceiver::SSI[RTPConstants::RTPMaxQualityLayers] [protected]

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

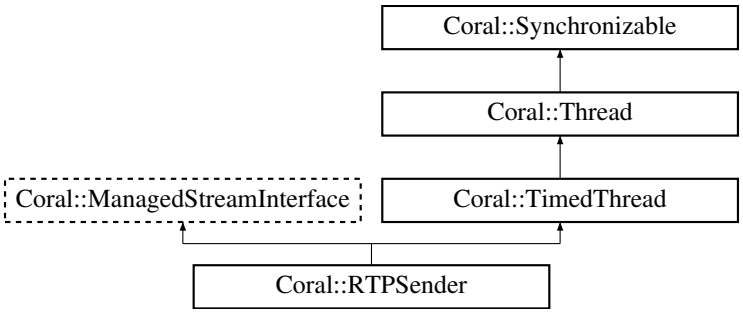
- rtpreceiver.h
- rtpreceiver.cc

6.82 Coral::RTPSender Class Reference

RTP Sender.

`#include <rtpsender.h>`

Inheritance diagram for Coral::RTPSender:



Public Member Functions

- RTPSender ()
- RTPSender (const card32 ssrc, EncoderInterface ∗encoder, Socket ∗senderSocket, const cardinal maxPacketSize=1500, BandwidthManager ∗bwManager=NULL)
- ∼RTPSender ()
- void init (const card32 ssrc, EncoderInterface ∗encoder, Socket ∗senderSocket, const cardinal maxPacketSize=1500, BandwidthManager ∗bwManager=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 () const
- card64 getPacketsSent () const
- void resetBytesSent ()
- void resetPacketsSent ()

Private Member Functions

- void timerEvent ()
- void update (const AbstractQoSDescription ∗aqd)

Private Attributes

- EncoderInterface * Encoder
- Socket * SenderSocket
- cardinal FramesPerSecond
- cardinal RenewCounter
- cardinal MaxPacketSize
- card32 SSRC
- card64 BytesSent
- card64 PacketsSent
- card64 TimeStamp
- card32 PayloadBytesSent
- card32 PayloadPacketsSent
- bool Pause
- bool TransmissionError
- InternetFlow Flow [RTPConstants::RTPMaxQualityLayers]
- card16 SequenceNumber [RTPConstants::RTPMaxQualityLayers]
- BandwidthManager * BandwidthMgr
- cardinal Bandwidth [RTPConstants::RTPMaxQualityLayers]
- double BufferDelay [RTPConstants::RTPMaxQualityLayers]
- TrafficShaper SenderReportBuffer
- TrafficShaper Shaper [RTPConstants::RTPMaxQualityLayers]

6.82.1 Detailed Description

RTP Sender.

This class implements an RTP sender based on TimedThread.

Author

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

6.82.2 Constructor & Destructor Documentation

6.82.2.1 Coral::RTPSender::RTPSender ( )

Default constructor. You have to initialize RTPSender by calling init(...) later!

See also

init

6.82.2.2 Coral::RTPSender::RTPSender (const card32 *ssrc*, EncoderInterface * *encoder*, Socket * *senderSocket*, const cardinal *maxPacketSize* = 1500, BandwidthManager * *bwManager* = NULL)

Constructor for new RTPSender. The new sender's thread has to be started by calling start()!

Parameters

ssrc	Sender's SSRC (see RFC 1889).
encoder	Encoder to get packets to send from.
senderSocket	Socket to write packets to.
maxPacketSize	Maximum packet size.
bwManager	Bandwidth manager.

See also

Thread::start

6.82.2.3 Coral::RTPSender::~RTPSender ( )

Destructor.

6.82.3 Member Function Documentation

6.82.3.1 card64 Coral::RTPSender::getBytesSent ( ) const [inline]

Get number of bytes sent.

Returns

Bytes sent.

6.82.3.2 cardinal Coral::RTPSender::getMaxPacketSize ( ) const [inline]

Get maximum packet size.

Returns

Maximum packet size.

6.82.3.3 card64 Coral::RTPSender::getPacketsSent ( ) const [inline]

Get number of packets sent.

Returns

Packets sent.

6.82.3.4 AbstractQoSDescription * Coral::RTPSender::getQoSDescription (const card64 *offset*) [virtual]

Implementation of ManagedStreamInterface's getQoSDescription().

See also

ManagedStreamInterface::getQoSDescription

Implements Coral::ManagedStreamInterface.

6.82.3.5 void Coral::RTPSender::init (const card32 *ssrc*, EncoderInterface * *encoder*, Socket * *senderSocket*, const cardinal *maxPacketSize* = 1500, BandwidthManager * *bwManager* = NULL)

Initialize new RTPSender. The new sender's thread has to be started by calling start()!

Parameters

ssrc	Sender's SSRC (see RFC 1889).
encoder	Encoder to get packets to send from.
senderSocket	Socket to write packets to.
maxPacketSize	Maximum packet size.
bwManager	Bandwidth manager.

See also

Thread::start

6.82.3.6 void Coral::RTPSender::lock ( ) [virtual]

Implementation of ManagedStreamInterface's lock().

See also

ManagedStreamInterface::lock

Implements Coral::ManagedStreamInterface.

6.82.3.7 bool Coral::RTPSender::paused ( ) const [inline]

Check, if transmission is paused.

Returns

true, if paused; false otherwise.

6.82.3.8 void Coral::RTPSender::resetBytesSent ( ) [inline]

Reset number of bytes sent.

6.82.3.9 void Coral::RTPSender::resetPacketsSent ( ) [inline]

Reset number of packets sent.

6.82.3.10 cardinal Coral::RTPSender::setMaxPacketSize (const cardinal *size*) [inline]

Set maximum packet size.

Parameters

size	Maximum packet size.

Returns

Maximum packet size set.

6.82.3.11 void Coral::RTPSender::setPause (const bool *on*) [inline]

Set pause on or off.

Parameters

on	true to set pause on; false to set pause off.

6.82.3.12 void Coral::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 Coral::TimedThread.

6.82.3.13 bool Coral::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.

6.82.3.14 void Coral::RTPSender::unlock ( ) [virtual]

Implementation of ManagedStreamInterface's unlock().

See also

ManagedStreamInterface::unlock

Implements Coral::ManagedStreamInterface.

6.82.3.15 void Coral::RTPSender::update (const AbstractQoSDescription * *aqd*) [private]

6.82.3.16 void Coral::RTPSender::updateQuality (const AbstractQoSDescription * *aqd*) [virtual]

Implementation of ManagedStreamInterface's updateQuality().

See also

ManagedStreamInterface::updateQuality

Implements Coral::ManagedStreamInterface.

6.82.4 Member Data Documentation

6.82.4.1 cardinal Coral::RTPSender::Bandwidth[RTPConstants::RTPMaxQuality-Layers] [private]

6.82.4.2 BandwidthManager* Coral::RTPSender::BandwidthMgr [private]

6.82.4.3 double Coral::RTPSender::BufferDelay[RTPConstants::RTPMaxQuality-Layers] [private]

6.82.4.4 card64 Coral::RTPSender::BytesSent [private]

6.82.4.5 EncoderInterface* Coral::RTPSender::Encoder [private]

6.82.4.6 InternetFlow Coral::RTPSender::Flow[RTPConstants::RTPMaxQuality-Layers] [private]

6.82.4.7 cardinal Coral::RTPSender::FramesPerSecond [private]

6.82.4.8 cardinal Coral::RTPSender::MaxPacketSize [private]

6.82.4.9 card64 Coral::RTPSender::PacketsSent [private]

6.82.4.10 bool Coral::RTPSender::Pause [private]

6.82.4.11 card32 Coral::RTPSender::PayloadBytesSent [private]

6.82.4.12 card32 Coral::RTPSender::PayloadPacketsSent [private]

6.82.4.13 cardinal Coral::RTPSender::RenewCounter [private]

6.82.4.14 TrafficShaper Coral::RTPSender::SenderReportBuffer [private]

6.82.4.15 Socket* Coral::RTPSender::SenderSocket [private]

6.82.4.16 card16 Coral::RTPSender::SequenceNumber[RTPConstants::RTPMaxQualityLayers] [private]

6.82.4.17 TrafficShaper Coral::RTPSender::Shaper[RTPConstants::RTPMaxQualityLayers] [private]

6.82.4.18 card32 Coral::RTPSender::SSRC [private]

6.82.4.19 card64 Coral::RTPSender::TimeStamp [private]

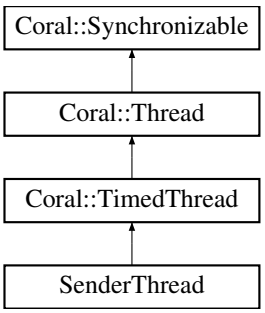
6.82.4.20 bool Coral::RTPSender::TransmissionError [private]

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

- rtpsender.h
- rtpsender.cc

6.83 SenderThread Class Reference

Inheritance diagram for SenderThread:



Public Member Functions

- SenderThread (TrafficShaper *socket, const card64 bandwidth, const cardinal packetSize=1024, const cardinal framesPerSecond=75)
- card64 getBytesSent ()
- void timerEvent ()

Public Attributes

- TrafficShaper * SenderSocket
- card64 Bandwidth
- card64 Sent
- cardinal BytesPerInterval
- cardinal PacketSize
- cardinal FramesPerSecond
- card16 SeqNum

6.83.1 Constructor & Destructor Documentation

6.83.1.1 SenderThread::SenderThread (TrafficShaper * *socket*, const card64 *bandwidth*, const cardinal *packetSize* = `1024`, const cardinal *framesPerSecond* = `75`)

6.83.2 Member Function Documentation

6.83.2.1 card64 SenderThread::getBytesSent ( ) `[inline]`

6.83.2.2 void SenderThread::timerEvent ( ) `[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 Coral::TimedThread.

6.83.3 Member Data Documentation

6.83.3.1 card64 SenderThread::Bandwidth

6.83.3.2 cardinal SenderThread::BytesPerInterval

6.83.3.3 cardinal SenderThread::FramesPerSecond

6.83.3.4 cardinal SenderThread::PacketSize

6.83.3.5 TrafficShaper* SenderThread::SenderSocket

6.83.3.6 card64 SenderThread::Sent

6.83.3.7 card16 SenderThread::SeqNum

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

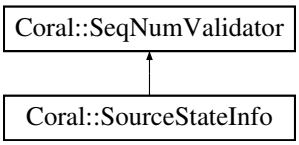
- t5.cc

6.84 Coral::SeqNumValidator Class Reference

Sequence Number Validator.

`#include <seqnumvalidator.h>`

Inheritance diagram for Coral::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

6.84.1 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

6.84.2 Member Enumeration Documentation

6.84.2.1 enum Coral::SeqNumValidator::ValidationResult

Enumerator:

Valid

SourceProbation

Jumped

Invalid

DuplicatePacket

InvalidSeqNum

6.84.3 Constructor & Destructor Documentation

6.84.3.1 Coral::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.

6.84.4 Member Function Documentation

6.84.4.1 double Coral::SeqNumValidator::calculateFractionLost ( )

Calculate and get fraction of packets lost.

Returns

Fraction lost.

6.84.4.2 double Coral::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

6.84.4.3 double Coral::SeqNumValidator::getJitter ( ) const [inline]

Get jitter.

Returns

Jitter.

6.84.4.4 card64 Coral::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.

6.84.4.5 card64 Coral::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.

6.84.4.6 card64 Coral::SeqNumValidator::getPacketsReceived ( ) const [inline]

Get number of packets received.

Returns

Number of packets received.

6.84.4.7 void Coral::SeqNumValidator::init (const card64 *sequenceNumber*) [inline, private]

6.84.4.8 void Coral::SeqNumValidator::reset ( )

Reset SeqNumValidator.

Reimplemented in Coral::SourceStateInfo.

6.84.4.9 SeqNumValidator::ValidationResult Coral::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.

6.84.5 Member Data Documentation

6.84.5.1 card64 Coral::SeqNumValidator::BadSeq [private]

6.84.5.2 card64 Coral::SeqNumValidator::BaseSeq [private]

6.84.5.3 card64 Coral::SeqNumValidator::Cycles [private]

6.84.5.4 card64 Coral::SeqNumValidator::ExpectedPrior [private]

Reimplemented in Coral::SourceStateInfo.

6.84.5.5 double Coral::SeqNumValidator::FractionLost [private]

Reimplemented in Coral::SourceStateInfo.

6.84.5.6 double Coral::SeqNumValidator::Jitter [private]

6.84.5.7 cardinal Coral::SeqNumValidator::MaxDropout [private]

6.84.5.8 cardinal Coral::SeqNumValidator::MaxMisorder [private]

6.84.5.9 card64 Coral::SeqNumValidator::MaxSeq [private]

6.84.5.10 cardinal Coral::SeqNumValidator::MinSequential [private]

6.84.5.11 card64 Coral::SeqNumValidator::PrevPacketArrivalTime [private]

6.84.5.12 card64 Coral::SeqNumValidator::PrevPacketTimeStamp [private]

6.84.5.13 card32 Coral::SeqNumValidator::Probation [private]

6.84.5.14 card64 Coral::SeqNumValidator::Received [private]

6.84.5.15 card64 Coral::SeqNumValidator::ReceivedPrior [private]

Reimplemented in Coral::SourceStateInfo.

6.84.5.16 card64 Coral::SeqNumValidator::SeqMod [private]

6.84.5.17 bool Coral::SeqNumValidator::Uninitialized [private]

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

- seqnumvalidator.h
- seqnumvalidator.cc

6.85 Coral::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]

6.85.1 Detailed Description

Trace Layer Configuration.

This class is a service level agreement (SLA).

Author

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

6.85.2 Constructor & Destructor Documentation

6.85.2.1 Coral::ServiceLevelAgreement::ServiceLevelAgreement ( )

Constructor.

6.85.2.2 Coral::ServiceLevelAgreement::~ServiceLevelAgreement ( )

Destructor.

6.85.3 Member Function Documentation

6.85.3.1 cardinal Coral::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).

6.85.3.2 bool Coral::ServiceLevelAgreement::load (const char * *fileName*)

Load configuration from file.

6.85.4 Member Data Documentation

6.85.4.1 cardinal Coral::ServiceLevelAgreement::BestEffort

6.85.4.2 DiffServClass Coral::ServiceLevelAgreement::Class[TrafficClassValues::MaxValues]

6.85.4.3 cardinal Coral::ServiceLevelAgreement::Classes

6.85.4.4 card64 Coral::ServiceLevelAgreement::TotalBandwidth

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

- servicelevelagreement.h
- servicelevelagreement.cc

6.86 Coral::SessionDescription Struct Reference

Session Description.

`#include <sessiondescription.h>`

Public Attributes

- cardinal SessionID
- cardinal Streams
- multimap < ManagedStreamInterface *, StreamDescription * > StreamSet
- card64 MinWantedBandwidth
- card64 MaxWantedBandwidth
- card64 TotalAllocatedBandwidth
- card64 AllocatedBandwidthArray [TrafficClassValues::MaxValues]
- int8 Priority
- bool MaximumReached

6.86.1 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

6.86.2 Member Data Documentation

6.86.2.1 card64 Coral::SessionDescription::AllocatedBandwidthArray[TrafficClassValues::MaxValues]

Allocated bandwidth for each class.

6.86.2.2 bool Coral::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.

6.86.2.3 card64 Coral::SessionDescription::MaxWantedBandwidth

Maximum session bandwidth.

6.86.2.4 card64 Coral::SessionDescription::MinWantedBandwidth

Minimum session bandwidth.

6.86.2.5 int8 Coral::SessionDescription::Priority

Session priority.

6.86.2.6 cardinal Coral::SessionDescription::SessionID

Session ID.

6.86.2.7 cardinal Coral::SessionDescription::Streams

Number of streams.

6.86.2.8 multimap<ManagedStreamInterface*,StreamDescription*> Coral::SessionDescription::StreamSet

Stream description set.

6.86.2.9 card64 Coral::SessionDescription::TotalAllocatedBandwidth

Total allocated bandwidth.

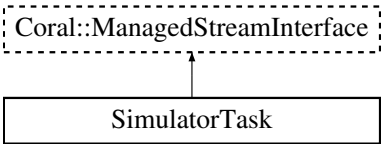
The documentation for this struct was generated from the following file:

- sessiondescription.h

6.87 SimulatorTask Class Reference

Simulator Task.

Inheritance diagram for SimulatorTask:



Public Member Functions

- SimulatorTask (BandwidthManager *bwManager, ServiceLevelAgreement *sla)
- ~SimulatorTask ()
- bool open (const char *name, const cardinal headerSize, const cardinal maxPacketSize)
- void close ()
- AbstractQoSDescription * getQoSDescription (const card64 offset)
- void updateQuality (const AbstractQoSDescription *aqd)
- void lock ()
- void unlock ()
- cardinal nextStep ()

Public Attributes

- double FrameRate
- TDTFMediaReader Reader
- TraceEncoder * Encoder
- BandwidthManager * BandwidthMgr
- ServiceLevelAgreement * SLA
- cardinal Layers
- card64 TotalBandwidth
- card64 TotalBandwidthDelay1
- card64 Bandwidth [RTPConstants::RTPMaxQualityLayers]
- card64 BandwidthDelay1 [RTPConstants::RTPMaxQualityLayers]

- double BufferDelayMS [RTPConstants::RTPMaxQualityLayers]
- cardinal BufferDelay [RTPConstants::RTPMaxQualityLayers]
- card8 TrafficClass [RTPConstants::RTPMaxQualityLayers]
- TrafficPolicer Policer [RTPConstants::RTPMaxQualityLayers]
- cardinal Flushes
- cardinal SessionID
- cardinal StreamID
- double CostFactor [TrafficClassValues::MaxValues]
- card64 Sent [TrafficClassValues::MaxValues]
- cardinal HeaderSize
- cardinal MaxPacketSize

Static Public Attributes

- static card64 SimulatorTime = 1000000000

Private Member Functions

- cardinal handleNextFrame ()

Private Attributes

- card64 LastCall

6.87.1 Detailed Description

Simulator Task.

This class is a simulator task.

Author

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

6.87.2 Constructor & Destructor Documentation

6.87.2.1 SimulatorTask::SimulatorTask (BandwidthManager * *bwManager*, ServiceLevelAgreement * *sla*)

Constructor.

6.87.2.2 SimulatorTask::~SimulatorTask ( )

Destructor.

6.87.3 Member Function Documentation

6.87.3.1 void SimulatorTask::close ( )

Close stream.

6.87.3.2 AbstractQoSDescription * SimulatorTask::getQoSDescription (const card64 *offset*) [virtual]

Implementation of ManagedStreamInterface's getQoSDescription().

See also

ManagedStreamInterface::getQoSDescription

Implements Coral::ManagedStreamInterface.

6.87.3.3 cardinal SimulatorTask::handleNextFrame ( ) [private]

6.87.3.4 void SimulatorTask::lock ( ) [virtual]

Implementation of ManagedStreamInterface's lock().

See also

ManagedStreamInterface::lock

Implements Coral::ManagedStreamInterface.

6.87.3.5 cardinal SimulatorTask::nextStep ( )

Do next simulation step.

Returns

0, if no operation was necessary; 1, if operation was successful; >1 otherwise (end of stream).

6.87.3.6 bool SimulatorTask::open (const char * *name*, const cardinal *headerSize*, const cardinal *maxPacketSize*)

Open stream.

Parameters

name	TDTF file name.
headerSize	Packet header size.
maxPacket-Size	Maximum packet size (e.g. 1500 for ethernet).

Returns

true, if operation was successful; false otherwise.

6.87.3.7 void SimulatorTask::unlock ( ) [virtual]

Implementation of ManagedStreamInterface's unlock().

See also

ManagedStreamInterface::unlock

Implements Coral::ManagedStreamInterface.

6.87.3.8 void SimulatorTask::updateQuality (const AbstractQoSDescription * *aqd*) [virtual]

Implementation of ManagedStreamInterface's updateQuality().

See also

ManagedStreamInterface::updateQuality

Implements Coral::ManagedStreamInterface.

6.87.4 Member Data Documentation

6.87.4.1 card64 SimulatorTask::Bandwidth[RTPConstants::RTPMaxQualityLayers]

6.87.4.2 card64 SimulatorTask::BandwidthDelay1[RTPConstants::RTPMaxQuality-Layers]

6.87.4.3 BandwidthManager* SimulatorTask::BandwidthMgr

6.87.4.4 cardinal SimulatorTask::BufferDelay[RTPConstants::RTPMaxQualityLayers]

6.87.4.5 double SimulatorTask::BufferDelayMS[RTPConstants::RTPMaxQualityLayers]

6.87.4.6 double SimulatorTask::CostFactor[TrafficClassValues::MaxValues]

6.87.4.7 TraceEncoder* SimulatorTask::Encoder

6.87.4.8 cardinal SimulatorTask::Flushes

6.87.4.9 double SimulatorTask::FrameRate

6.87.4.10 cardinal SimulatorTask::HeaderSize

6.87.4.11 card64 SimulatorTask::LastCall [private]

6.87.4.12 cardinal SimulatorTask::Layers

6.87.4.13 cardinal SimulatorTask::MaxPacketSize

6.87.4.14 TrafficPolicer SimulatorTask::Policer[RTPConstants::RTPMaxQualityLayers]

6.87.4.15 TDTFMediaReader SimulatorTask::Reader

6.87.4.16 card64 SimulatorTask::Sent[TrafficClassValues::MaxValues]

6.87.4.17 cardinal SimulatorTask::SessionID

6.87.4.18 card64 SimulatorTask::SimulatorTime = 1000000000 [static]

6.87.4.19 ServiceLevelAgreement* SimulatorTask::SLA

6.87.4.20 cardinal SimulatorTask::StreamID

6.87.4.21 card64 SimulatorTask::TotalBandwidth

6.87.4.22 card64 SimulatorTask::TotalBandwidthDelay1

6.87.4.23 card8 SimulatorTask::TrafficClass[RTPConstants::RTPMaxQualityLayers]

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

- tsimulator.cc

6.88 Coral::Socket Class Reference

Socket.

```
#include <socket.h>
```

Public Types

- enum SocketCommunicationDomain { UndefinedSocketCommunicationDomain = -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 }

Public Member Functions

- Socket ()
- Socket (const SocketCommunicationDomain communicationDomain, const SocketType socketType, const SocketProtocol socketProtocol=Default)
- virtual ~Socket ()
- bool create (const SocketCommunicationDomain communicationDomain=IP, const SocketType socketType=TCP, const SocketProtocol socketProtocol=Default)
- void close ()
- void shutdown (const cardinal shutdownLevel)
- card64 getBytesSent () const
- card64 getBytesReceived () const
- void resetBytesSent ()
- void resetBytesReceived ()
- bool ready () const
- bool bind (const SocketAddress &address=InternetAddress())
- bool listen (const cardinal backlog=5)
- Socket * accept ()
- bool connect (const SocketAddress &address, const card8 trafficClass=0)
- integer getLastError ()
- integer getSocketOption (const cardinal level, const cardinal optionNumber, void *optionValue, socklen_t *optionLength) const
- cardinal getSoLinger () const
- bool getSoReuseAddress () const
- bool getSoBroadcast () const
- bool getTCPNoDelay () const
- bool getBlockingMode ()
- integer setSocketOption (const cardinal level, const cardinal optionNumber, const void *optionValue, const socklen_t optionLength)
- void setSoLinger (const bool on, const cardinal linger)
- void setSoReuseAddress (const bool on)
- void setSoBroadcast (const bool on)
- void setTCPNoDelay (const bool on)
- void setBlockingMode (const bool on)
- card32 getSendFlowLabel () const

- card8 getSendTrafficClass () const
- card32 getReceivedFlowLabel () const
- card8 getReceivedTrafficClass () const
- virtual ssize_t sendTo (const void *buffer, const size_t length, const cardinal flags, const SocketAddress &receiver, const card8 trafficClass=0)
- virtual ssize_t send (const void *buffer, const size_t length, const cardinal flags=0, const card8 trafficClass=0)
- virtual ssize_t receiveFrom (void *buffer, const size_t length, SocketAddress &sender, const cardinal flags=0)
- virtual ssize_t receive (void *buffer, const size_t length, const cardinal flags=0)
- virtual ssize_t read (void *buffer, const size_t length)
- virtual ssize_t write (const void *buffer, const size_t length)
- integer fcntl (const integer cmd, const long arg=0)
- integer ioctl (const integer request, const void *argp)
- bool getSocketAddress (SocketAddress &address) const
- bool getPeerAddress (SocketAddress &address) const
- InternetFlow allocFlow (const InternetAddress &address, const card32 flowLabel=0, const card8 shareLevel=IPV6_FL_S_PROCESS)
- void freeFlow (InternetFlow &flow)
- bool renewFlow (InternetFlow &flow, const cardinal expires, const cardinal linger=6)
- bool renewFlow (const cardinal expires, const cardinal linger=6)
- virtual void setTrafficConstraint (const card8 trafficClass, const card64 bandwidth, const double bufferDelay)
- virtual void flush ()
- int getSystemSocketDescriptor () const

Static Public Member Functions

- static bool bindInternetSocketPair (Socket &senderSocket, Socket &receiverSocket, const InternetAddress &receiver=InternetAddress())

Static Public Attributes

- static const cardinal MinAutoSelectPort = 16384
- static const cardinal MaxAutoSelectPort = 65535

Protected Member Functions

- void init ()
- bool setTOS (const card8 trafficClass)
- ssize_t recvFrom (int fd, void *buf, const size_t len, const integer flags, struct sockaddr *addr, size_t *addrlen)

Protected Attributes

- card64 BytesSent
- card64 BytesReceived
- card32 SendFlow
- card32 ReceivedFlow
- cardinal Backlog
- cardinal LastError
- int SocketDescriptor
- sockaddr * Destination
- SocketCommunicationDomain CommunicationDomain
- SocketType Type
- SocketProtocol Protocol

Friends

- class TrafficShaper

6.88.1 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

6.88.2 Member Enumeration Documentation

6.88.2.1 enum Coral::Socket::SocketCommunicationDomain

Enumerator:

UndefinedSocketCommunicationDomain

IP

IPv4

IPv6

Unix

6.88.2.2 enum Coral::Socket::SocketProtocol

Enumerator:

UndefinedSocketProtocol

Default

ICMPv4

ICMPv6

6.88.2.3 enum Coral::Socket::SocketType

Enumerator:

UndefinedSocketType

UDP

Datagram

TCP

Stream

Raw

RDM

SeqPacket

6.88.3 Constructor & Destructor Documentation

6.88.3.1 Coral::Socket::Socket ( )

Constructor.

6.88.3.2 Coral::Socket::Socket (const SocketCommunicationDomain *communicationDomain,* const SocketType *socketType,* const SocketProtocol *socketProtocol* = Default)

Constructor for a new socket. For automatic usage of IPv6 when available, set communication domain to IP. Use IPv4/IPv6 only if a special protocol version is necessary! The creation success can be checked using ready() method.

Parameters

communicationDomain	Communication domain (e.g. IP).
socketType	Socket type (e.g. TCP, UDP).
socketProtocol	Socket protocol (e.g. Default).

See also

ready

6.88.3.3 Coral::Socket::~Socket ( ) [virtual]

Destructor.

6.88.4 Member Function Documentation

6.88.4.1 Socket * Coral::Socket::accept ( )

Accept a connection.

Returns

New socket.

6.88.4.2 InternetFlow Coral::Socket::allocFlow (const InternetAddress & *address*, const card32 *flowLabel* = 0, const card8 *shareLevel* = IPV6_FL_S_PROCESS)

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.

6.88.4.3 bool Coral::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.

6.88.4.4 bool Coral::Socket::bindInternetSocketPair (Socket & *senderSocket,* Socket & *receiverSocket,* const InternetAddress & *receiver* = InternetAddress ()) [static]

Bind a pair of internet 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

senderSocket	First socket.
receiverSocket	Second socket.
receiver	Address (e.g ipv6-localhost:0) or NULL for Any address.

6.88.4.5 void Coral::Socket::close ( )

Close socket.

6.88.4.6 bool Coral::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.

6.88.4.7 bool Coral::Socket::create (const SocketCommunicationDomain *communicationDomain* = IP, const SocketType *socketType* = TCP, const SocketProtocol *socketProtocol* = Default)

Close existing socket and create new socket. For automatic usage of IPv6 when available, set communication domain to IP. Use IPv4/IPv6 only if a special protocol version is necessary!

Parameters

communication-Domain	Communication domain (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.

6.88.4.8 integer Coral::Socket::fcntl (const integer *cmd,* const long *arg* = 0) [inline]

Wrapper for fcntl().

Parameters

cmd	Command.
arg	Argument.

Returns

Result of fcntl() call.

6.88.4.9 void Coral::Socket::flush ( ) [virtual]

Flush traffic shaper buffer. Note: This functionality has to be implemented by subclasses!

6.88.4.10 void Coral::Socket::freeFlow (InternetFlow & *flow*)

Free a flow.

Parameters

flow	Flow to be freed.

6.88.4.11 bool Coral::Socket::getBlockingMode ( )

Check, if blocking mode is on.

Returns

true, if blocking mode is on; false otherwise.

6.88.4.12 card64 Coral::Socket::getBytesReceived ( ) const [inline]

Get number of bytes received.

Returns

Number of bytes received.

6.88.4.13 card64 Coral::Socket::getBytesSent ( ) const [inline]

Get number of bytes sent.

Returns

Number of bytes sent.

6.88.4.14 integer Coral::Socket::getLastError ( ) [inline]

Get last error code. It will be reset to 0 after copying.

Returns

Last error code.

6.88.4.15 bool Coral::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

6.88.4.16 card32 Coral::Socket::getReceivedFlowLabel ( ) const [inline]

Get last received flow label.

Returns

Last received flow label or 0, if there is no flow label.

6.88.4.17 card8 Coral::Socket::getReceivedTrafficClass ( ) const [inline]

Get last received traffic class.

Returns

Last received traffic class or 0, if there is no traffic class.

6.88.4.18 card32 Coral::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

6.88.4.19 card8 Coral::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

6.88.4.20 bool Coral::Socket::getSoBroadcast ( ) const

Get SO_BROADCAST option of socket.

Returns

SO_BROADCAST value.

6.88.4.21 bool Coral::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

6.88.4.22 integer Coral::Socket::getSocketOption (const cardinal *level,* const cardinal *optionNumber,* void * *optionValue,* socklen_t * *optionLength*) const [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().

6.88.4.23 cardinal Coral::Socket::getSoLinger ( ) const

Get SO_LINGER option of socket.

Returns

SO_LINGER value.

6.88.4.24 bool Coral::Socket::getSoReuseAddress ( ) const

Get SO_REUSEADDR option of socket.

Returns

SO_REUSEADDR value.

6.88.4.25 int Coral::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.

6.88.4.26 bool Coral::Socket::getTCPNoDelay ( ) const

Get TCP_NODELAY option of socket.

Returns

TCP_NODELAY value.

6.88.4.27 void Coral::Socket::init ( ) [protected]

6.88.4.28 integer Coral::Socket::ioctl (const integer *request*, const void * *argp*) [inline]

Wrapper for ioctl().

Parameters

request	Request.
argp	Argument.

Returns

Result of ioctl() call.

6.88.4.29 bool Coral::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.

6.88.4.30 ssize_t Coral::Socket::read (void * *buffer*, const size_t *length*) [virtual]

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.

6.88.4.31 bool Coral::Socket::ready ( ) const [inline]

Check, if socket is ready.

Returns

true, if socket is ready; false otherwise.

6.88.4.32 ssize_t Coral::Socket::receive (void * *buffer*, const size_t *length*, const cardinal *flags* = 0) [virtual]

Wrapper for recv().

Parameters

buffer	Buffer to read data to.
length	Maximum length of data to be received.
flags	Flags for recv().

Returns

Bytes read or error code < 0.

6.88.4.33 ssize_t Coral::Socket::receiveFrom (void * *buffer*, const size_t *length*, SocketAddress & *sender*, const cardinal *flags* = 0) [virtual]

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 recvfrom().

Returns

Bytes received or error code < 0.

6.88.4.34 ssize_t Coral::Socket::recvFrom (int *fd*, void * *buf*, const size_t *len*, const integer *flags*, struct sockaddr * *addr*, size_t * *addrlen*) [protected]

6.88.4.35 bool Coral::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.

6.88.4.36 bool Coral::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.

6.88.4.37 void Coral::Socket::resetBytesReceived ( ) [inline]

Reset number of bytes received.

6.88.4.38 void Coral::Socket::resetBytesSent ( ) [inline]

Reset number of bytes sent.

6.88.4.39 ssize_t Coral::Socket::send (const void * *buffer*, const size_t *length*, const cardinal *flags* = 0, const card8 *trafficClass* = 0) [virtual]

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.

6.88.4.40 ssize_t Coral::Socket::sendTo (const void * *buffer*, const size_t *length*, const cardinal *flags*, const SocketAddress & *receiver*, const card8 *trafficClass* = 0) [virtual]

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.

Returns

Bytes sent or error code < 0.

6.88.4.41 void Coral::Socket::setBlockingMode (const bool *on*)

Set blocking mode.

Parameters

on	True to set blocking mode, false to unset.

6.88.4.42 void Coral::Socket::setSoBroadcast (const bool *on*)

Set SO_BROADCAST option of socket.

Parameters

on	true to set SO_BROADCAST on; false otherwise.

6.88.4.43 integer Coral::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().

6.88.4.44 void Coral::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.

6.88.4.45 void Coral::Socket::setSoReuseAddress (const bool *on*)

Set SO_REUSEADDR option of socket.

Parameters

on	true to set SO_REUSEADDR on; false otherwise.

6.88.4.46 void Coral::Socket::setTCPNoDelay (const bool *on*)

Set TCP_NODELAY option of socket.

Parameters

on	true to set TCP_NODELAY on; false otherwise.

6.88.4.47 bool Coral::Socket::setTOS (const card8 *trafficClass*) [protected]

6.88.4.48 void Coral::Socket::setTrafficConstraint (const card8 *trafficClass,* const card64 *bandwidth,* const double *bufferDelay*) [virtual]

Set traffic constraint to given byte rate. Packets exceeding the given constraint will be dropped. Note: This functionality has to be implemented by subclasses!

Parameters

trafficClass	Traffic class.
bandwidth	Bandwidth.
bufferDelay	Maximum buffer delay in microseconds.

6.88.4.49 void Coral::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.

6.88.4.50 ssize_t Coral::Socket::write (const void * *buffer*, const size_t *length*) [virtual]

Wrapper for write().

Parameters

buffer	Buffer with data to write
length	Length of data to write

Returns

Bytes sent or error code < 0.

6.88.5 Friends And Related Function Documentation

6.88.5.1 friend class TrafficShaper [friend]

6.88.6 Member Data Documentation

6.88.6.1 cardinal Coral::Socket::Backlog [protected]

6.88.6.2 card64 Coral::Socket::BytesReceived [protected]

6.88.6.3 card64 Coral::Socket::BytesSent [protected]

6.88.6.4 SocketCommunicationDomain Coral::Socket::CommunicationDomain [protected]

6.88.6.5 sockaddr* Coral::Socket::Destination [protected]

6.88.6.6 cardinal Coral::Socket::LastError [protected]

6.88.6.7 const cardinal Coral::Socket::MaxAutoSelectPort = 65535 [static]

Maximum port number for bind()'s automatic port selection.

See also

bind

6.88.6.8 const cardinal Coral::Socket::MinAutoSelectPort = 16384 [static]

Minimum port number for bind()'s automatic port selection.

See also

bind

6.88.6.9 SocketProtocol Coral::Socket::Protocol [protected]

6.88.6.10 card32 Coral::Socket::ReceivedFlow [protected]

6.88.6.11 card32 Coral::Socket::SendFlow [protected]

6.88.6.12 int Coral::Socket::SocketDescriptor [protected]

6.88.6.13 SocketType Coral::Socket::Type [protected]

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

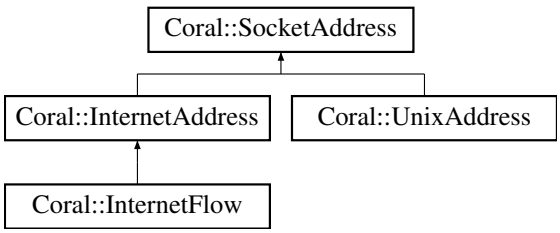
- socket.h
- socket.cc

6.89 Coral::SocketAddress Class Reference

Socket Address.

`#include <socketaddress.h>`

Inheritance diagram for Coral::SocketAddress:



Public Member Functions

- virtual void reset ()=0
- virtual bool isValid () const =0
- virtual String getAddressString () const =0
- virtual cardinal getSystemAddress (sockaddr *buffer, const socklen_t length, const cardinal type) const =0
- virtual bool setSystemAddress (sockaddr *address, const socklen_t length)=0

Static Public Attributes

- static const cardinal MaxSockLen

6.89.1 Detailed Description

Socket Address.

This class is an interface for a socket address.

Author

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

6.89.2 Member Function Documentation

6.89.2.1 virtual String Coral::SocketAddress::getAddressString ( ) const [pure virtual]

Get address string.

Returns

Address string.

Implemented in Coral::InternetAddress, Coral::UnixAddress, and Coral::InternetFlow.

6.89.2.2 virtual cardinal Coral::SocketAddress::getSystemAddress (sockaddr * *buffer*, const socklen_t *length*, const cardinal *type*) const [pure virtual]

Get system's sockaddr structure for the address.

Parameters

buffer	Buffer to write sockaddr to.
length	Length of buffer.
type	AF_INET or AF_INET6.

Returns

Length of written sockaddr structure.

Implemented in Coral::InternetAddress, and Coral::InternetFlow.

6.89.2.3 virtual bool Coral::SocketAddress::isValid ( ) const [pure virtual]

Check, if address is valid.

Returns

true, if address is valid; false otherwise.

Implemented in Coral::InternetAddress, and Coral::UnixAddress.

6.89.2.4 virtual void Coral::SocketAddress::reset ( ) [pure virtual]

Reset address.

Implemented in Coral::InternetAddress, Coral::UnixAddress, and Coral::InternetFlow.

6.89.2.5 virtual bool Coral::SocketAddress::setSystemAddress (sockaddr * *address*, const socklen_t *length*) [pure virtual]

Initialize the internet address from the system's sockaddr structure. The sockaddr structure may be sockaddr_in (AF_INET) or sockaddr_in6 (AF_INET6).

Parameters

address	sockaddr.
length	Length of sockaddr (sizeof(sockaddr_in) or sizeof(sockaddr_in6)).

Implemented in Coral::InternetAddress, and Coral::InternetFlow.

6.89.3 Member Data Documentation

6.89.3.1 const cardinal Coral::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 file:

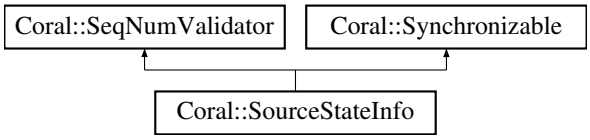
- socketaddress.h

6.90 Coral::SourceStateInfo Class Reference

Source State Info.

`#include <sourcestateinfo.h>`

Inheritance diagram for Coral::SourceStateInfo:



Public Member Functions

- SourceStateInfo ()
- 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

6.90.1 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

6.90.2 Constructor & Destructor Documentation

6.90.2.1 Coral::SourceStateInfo::SourceStateInfo ( )

Constructor.

6.90.3 Member Function Documentation

6.90.3.1 card32 Coral::SourceStateInfo::calculateDLSR ( ) const

Calculate delay since last sender report time stamp using current time.

Returns

DLSR.

6.90.3.2 card32 Coral::SourceStateInfo::getLSR ( ) const [inline]

Get last sender report time stamp.

Returns

LSR.

6.90.3.3 card32 Coral::SourceStateInfo::getSSRC ( ) const [inline]

Get SSRC.

Returns

SSRC.

6.90.3.4 void Coral::SourceStateInfo::reset ( )

Reset.

Reimplemented from Coral::SeqNumValidator.

6.90.3.5 void Coral::SourceStateInfo::setLSR (const card32 *lsr*) [inline]

Set last sender report time stamp.

Parameters

lsr	LSR.

6.90.3.6 void Coral::SourceStateInfo::setSSRC (card32 *ssrc*) [inline]

Set SSRC.

Returns

SSRC.

6.90.4 Member Data Documentation

6.90.4.1 card32 Coral::SourceStateInfo::ExpectedPrior [private]

Reimplemented from Coral::SeqNumValidator.

6.90.4.2 card32 Coral::SourceStateInfo::FractionLost [private]

Reimplemented from Coral::SeqNumValidator.

6.90.4.3 card32 Coral::SourceStateInfo::LSR [private]

6.90.4.4 card64 Coral::SourceStateInfo::LSRUpdateTimeStamp [private]

6.90.4.5 card32 Coral::SourceStateInfo::ReceivedPrior [private]

Reimplemented from Coral::SeqNumValidator.

6.90.4.6 card32 Coral::SourceStateInfo::SSRC [private]

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

- sourcestateinfo.h
- sourcestateinfo.cc

6.91 Coral::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

Static Public Attributes

- static const cardinal MaxRUEntries = 256

Private Member Functions

- bool calculatePossibleLayerClassMappings (ResourceUtilizationPoint &rup, const ServiceLevelAgreement *sla, const AbstractQoSDescription *aqd)

6.91.1 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

6.91.2 Constructor & Destructor Documentation

6.91.2.1 Coral::StreamDescription::StreamDescription ( )

Constructor.

6.91.2.2 Coral::StreamDescription::~StreamDescription ( )

Destructor.

6.91.3 Member Function Documentation

6.91.3.1 bool Coral::StreamDescription::calculatePossibleLayerClassMappings (ResourceUtilizationPoint & *rup,* const ServiceLevelAgreement * *sla,* const AbstractQoSDescription * *aqd*) [private]

6.91.3.2 void Coral::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.

6.91.3.3 bool Coral::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).

6.91.4 Member Data Documentation

6.91.4.1 cardinal Coral::StreamDescription::BufferFlushes

Number of buffer flushes.

6.91.4.2 cardinal Coral::StreamDescription::CompleteRemappings

Number of complete remappings forced by this stream.

6.91.4.3 double Coral::StreamDescription::CurrentCostPerSecond

Current bandwidth cost per second.

6.91.4.4 card64 Coral::StreamDescription::CurrentLayerClassBandwidth[RTP-Constants::RTPMaxQualityLayers]

Current layer's allocated bandwidth currently used.

6.91.4.5 cardinal Coral::StreamDescription::CurrentLayerClassNumber[RTP-Constants::RTPMaxQualityLayers]

Layer's allocated DiffServ class number currently used.

6.91.4.6 ResourceUtilizationPoint Coral::StreamDescription::CurrentQuality

Current quality.

6.91.4.7 cardinal Coral::StreamDescription::Inits

Number of init() calls.

See also

init

6.91.4.8 ManagedStreamInterface* Coral::StreamDescription::Interface

Stream's manager interface.

6.91.4.9 card64 Coral::StreamDescription::LastInitDuration

Duration of last init() call.

6.91.4.10 double Coral::StreamDescription::LastUtilization

Last *complete* remapping's utilization.

6.91.4.11 cardinal Coral::StreamDescription::Layers

Number of layers.

6.91.4.12 bool Coral::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.

6.91.4.13 const cardinal Coral::StreamDescription::MaxRUEntries = 256 [static]

Maximum number of entries in the list.

6.91.4.14 double Coral::StreamDescription::MeasuredTransferDelay[TrafficClassValues::MaxValues]

Smoothed measured transfer delay from ICMP echo replies for each DiffServ class.

6.91.4.15 double Coral::StreamDescription::NewCostPerSecond

New remapping's bandwidth cost per second.

6.91.4.16 card64 Coral::StreamDescription::NewLayerClassBandwidth[RTPConstants::RTPMaxQualityLayers]

New remapping's layer's allocated bandwidth.

6.91.4.17 cardinal Coral::StreamDescription::NewLayerClassNumber[RTPConstants::RTPMaxQualityLayers]

New remapping's layer's allocated DiffServ class number.

6.91.4.18 ResourceUtilizationPoint Coral::StreamDescription::NewQuality

New remapping's quality.

6.91.4.19 card64 Coral::StreamDescription::NextInterval

Time to next interval in microseconds.

6.91.4.20 cardinal Coral::StreamDescription::PartialRemappings

Number of successful partial remappings.

6.91.4.21 AbstractQoSDescription* Coral::StreamDescription::QoSDescription

Stream's AbstractQoSDescription.

6.91.4.22 double Coral::StreamDescription::ReportedJitter[RTPConstants::RTP-MaxQualityLayers]

Smoothed reported jitter for each layer (via RTCP reports).

6.91.4.23 double Coral::StreamDescription::ReportedLossRate[RTPConstants::R-TPMaxQualityLayers]

Smoothed received loss rate for each layer (via RTCP reports).

6.91.4.24 card64 Coral::StreamDescription::ReservationTimeStamp

Reservation time stamp.

6.91.4.25 InternetAddress Coral::StreamDescription::RoundTripTimeDestination

Round trip time measurement destination.

6.91.4.26 cardinal Coral::StreamDescription::RUEntries

Number of entries in resource/utilization list.

6.91.4.27 ResourceUtilizationPoint Coral::StreamDescription::RUList[MaxRU-Entries]

Resource/utilization list.

6.91.4.28 SessionDescription* Coral::StreamDescription::Session

Session description.

6.91.4.29 card64 Coral::StreamDescription::StreamID

Stream ID.

6.91.4.30 double Coral::StreamDescription::TotalBandwidthUsage

Total bandwidth usage of this stream.

6.91.4.31 double Coral::StreamDescription::TotalCost

Total cost of this stream.

6.91.4.32 cardinal Coral::StreamDescription::TotalReservationUpdates

Total reservation updates.

6.91.4.33 double Coral::StreamDescription::TotalRuntime

Total runtime of this stream.

6.91.4.34 double Coral::StreamDescription::TotalUtilization

Total utilization of this stream.

6.91.4.35 bool Coral::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

6.92 StreamEntry Struct Reference

Public Member Functions

- int operator< (const StreamEntry &s) const
- int operator> (const StreamEntry &s) const
- int operator== (const StreamEntry &s) const

Public Attributes

- card64 ID
- String Title
- String File1
- String File2
- cardinal Session
- cardinal Flushes
- ofstream ∗ DataStream1
- ofstream ∗ DataStream2
- bool Removed

6.92.1 Member Function Documentation

6.92.1.1 int StreamEntry::operator< (const StreamEntry & *s*) const [inline]

6.92.1.2 int StreamEntry::operator== (const StreamEntry & *s*) const [inline]

6.92.1.3 int StreamEntry::operator> (const StreamEntry & *s*) const [inline]

6.92.2 Member Data Documentation

6.92.2.1 ofstream* StreamEntry::DataStream1

6.92.2.2 ofstream* StreamEntry::DataStream2

6.92.2.3 String StreamEntry::File1

6.92.2.4 String StreamEntry::File2

6.92.2.5 cardinal StreamEntry::Flushes

6.92.2.6 card64 StreamEntry::ID

6.92.2.7 bool StreamEntry::Removed

6.92.2.8 cardinal StreamEntry::Session

6.92.2.9 String StreamEntry::Title

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

- loganalyzer.cc

6.93 String Class Reference

String.

```
#include <strings.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

6.93.1 Detailed Description

String.

This class implements the String datatype.

Author

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

6.93.2 Constructor & Destructor Documentation

6.93.2.1 String::String ( )

Constructor for an empty string.

6.93.2.2 String::String (const String & *string*)

Constructor for a copy of a string.

Parameters

string	String to be copied.

6.93.2.3 String::String (const char * *string*)

Constructor for a copy of a string.

Parameters

string	String to be copied.

6.93.2.4 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.

6.93.2.5 String::String (const cardinal *value*)

Constructor for a string from a number.

Parameters

value	Number.

6.93.2.6 String::~String ( )

Destructor.

6.93.3 Member Function Documentation

6.93.3.1 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.

6.93.3.2 const char* String::getData ( ) const [inline]

Get string data.

Returns

String data.

6.93.3.3 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.

6.93.3.4 bool String::isNull ( ) const [inline]

Check, if string is NULL.

Returns

true, if string is NULL; false otherwise.

6.93.3.5 String String::left (const cardinal *maxChars*) const

Get left part of string.

Parameters

maxChars	Maximum number of characters to be copied.

Returns

String.

6.93.3.6 cardinal String::length ( ) const [inline]

Get string length.

Returns

Length in bytes.

6.93.3.7 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.

6.93.3.8 String String::mid (const cardinal *start*) const [inline]

Get part from start to end of string.

Parameters

start	Start position in String.

Returns

String.

6.93.3.9 int String::operator!= (const String & *string*) const [inline]

Implementation of != operator.

6.93.3.10 int String::operator< (const String & *string*) const [inline]

Implementation of < operator.

6.93.3.11 int String::operator<= (const String & *string*) const [inline]

Implementation of <= operator.

6.93.3.12 String & String::operator= (const String & *string*)

Implementation of = operator.

6.93.3.13 String & String::operator= (const char * *string*)

Implementation of = operator.

6.93.3.14 String & String::operator= (const cardinal *value*)

Implementation of = operator.

6.93.3.15 int String::operator== (const String & *string*) const [inline]

Implementation of == operator.

6.93.3.16 int String::operator> (const String & *string*) const [inline]

Implementation of > operator.

6.93.3.17 int String::operator>= (const String & *string*) const [inline]

Implementation of >= operator.

6.93.3.18 char String::operator[] (const int *index*) const [inline]

Implementation of [] operator.

6.93.3.19 String String::right (const cardinal *maxChars*) const

Get right part of string.

Parameters

maxChars	Maximum number of characters to be copied.

Returns

String.

6.93.3.20 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.

6.93.3.21 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.

6.93.3.22 void String::setData (char * *string*) [inline, private]

6.93.3.23 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.

6.93.3.24 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.

6.93.3.25 static cardinal String::stringLength (const char * *string*) [inline, static]

Compute length of a string.

Parameters

string	String.

Returns

Length.

6.93.3.26 String String::stripWhiteSpace ( ) const

Get string with spaces from beginning and end of the string removed.

Returns

New string.

6.93.3.27 String String::toLower ( ) const

Get lowercase string from string.

Returns

Lowercase string.

6.93.3.28 String String::toUpper ( ) const

Get uppercase string from string.

Returns

Uppercase string.

6.93.4 Member Data Documentation

6.93.4.1 char* String::Data [private]

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

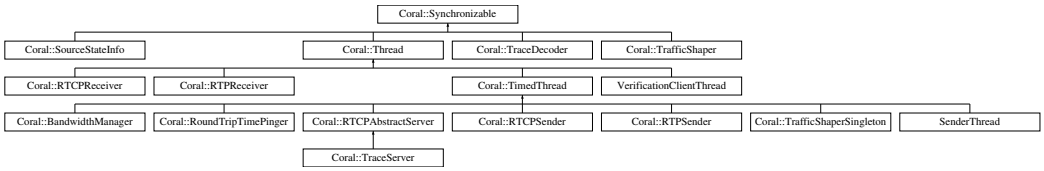
- strings.h
- strings.cc

6.94 Coral::Synchronizable Class Reference

Synchronizable.

`#include <synchronizable.h>`

Inheritance diagram for Coral::Synchronizable:



Public Member Functions

- Synchronizable ()
- ~Synchronizable ()
- void synchronized ()
- bool synchronizedTry ()
- void unsynchronized ()
- void resynchronize ()
- void synchronized_debug (const char *file, const cardinal line)
- void unsynchronized_debug (const char *file, const cardinal line)
- bool synchronizedTry_debug (const char *file, const cardinal line)
- void resynchronize_debug (const char *file, const cardinal line)

Private Attributes

- pthread_mutex_t Mutex

6.94.1 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

6.94.2 Constructor & Destructor Documentation

6.94.2.1 Coral::Synchronizable::Synchronizable ( )

Constructor

6.94.2.2 Coral::Synchronizable::~Synchronizable ( )

Destructor

6.94.3 Member Function Documentation

6.94.3.1 void Coral::Synchronizable::resynchronize ( )

Do reinitialization of Synchronizable.

6.94.3.2 void Coral::Synchronizable::resynchronize_debug (const char * *file*, const cardinal *line*)

Debug version of resynchronize. This will print PID, file name and line number, followed by debug information.

Parameters

file	File name.
line	Line number.

See also

resynchronize

6.94.3.3 void Coral::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

6.94.3.4 void Coral::Synchronizable::synchronized_debug (const char * *file*, const cardinal *line*)

Debug version of synchronized. This will print PID, file name and line number, followed by debug information.

Parameters

file	File name.
line	Line number.

See also

synchronized

6.94.3.5 bool Coral::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

6.94.3.6 bool Coral::Synchronizable::synchronizedTry_debug (const char * *file*, const cardinal *line*)

Debug version of synchronizedTry. This will print PID, file name and line number, followed by debug information.

Parameters

file	File name.
line	Line number.

See also

synchronizedTry

6.94.3.7 void Coral::Synchronizable::unsynchronized ( ) [inline]

unsynchronized() ends a synchronized block, which has begun by synchronized().

See also

synchronized

6.94.3.8 void Coral::Synchronizable::unsynchronized_debug (const char * *file*, const cardinal *line*)

Debug version of unsynchronized. This will print PID, file name and line number, followed by debug information.

Parameters

file	File name.
line	Line number.

See also

unsynchronized

6.94.4 Member Data Documentation

6.94.4.1 pthread_mutex_t Coral::Synchronizable::Mutex [private]

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

- synchronizable.h
- synchronizable.cc

6.95 Task Struct Reference

Public Attributes

- card64 LastDump
- SimulatorTask * Simulator
- StreamDescription * Stream
- GNUPlotData GNUplot
- ofstream Description
- cardinal Layers
- cardinal SessionID
- cardinal StreamID
- double MaxTransferDelay
- String MediaName

6.95.1 Member Data Documentation

6.95.1.1 ofstream Task::Description

6.95.1.2 GNUPlotData Task::GNUplot

6.95.1.3 card64 Task::LastDump

6.95.1.4 cardinal Task::Layers

6.95.1.5 double Task::MaxTransferDelay

6.95.1.6 String Task::MediaName

6.95.1.7 cardinal Task::SessionID

6.95.1.8 SimulatorTask* Task::Simulator

6.95.1.9 StreamDescription* Task::Stream

6.95.1.10 cardinal Task::StreamID

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

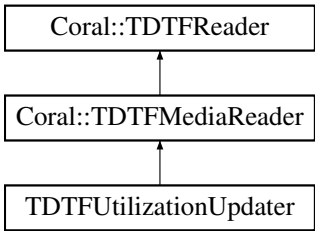
- tsimulator.cc

6.96 Coral::TDTFMediaReader Class Reference

TDTF Media Reader.

`#include <tdtfmediareader.h>`

Inheritance diagram for Coral::TDTFMediaReader:



Public Member Functions

- TDTFMediaReader ()
- bool open (const char *name, const bool readWrite=false, const bool ru-Required=false)
- void close ()
- bool ready () const
- void getMediaInfo (MediaInfo &mediaInfo) const
- MediaError getErrorCode () const
- card64 getPosition () const
- card64 getMaxPosition () const
- void setPosition (const card64 position)
- double getFrameRate () const
- double setFrameRate (const double frameRate)
- bool checkInterval (card64 &time, bool &newRUList)
- cardinal getNextBlock (const cardinal layers, cardinal *frameIDArray, cardinal *frameSizeArray)

Private Attributes

- const IntervalHeader * Interval
- const ResourceUtilizationHeader * RUHeader
- card64 Position
- card64 MaxPosition
- double FrameRate
- MediaError ErrorCode

6.96.1 Detailed Description

TDTF Media Reader.

This class is a media reader for TDTF trace files.

Author

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

6.96.2 Constructor & Destructor Documentation

6.96.2.1 Coral::TDTFMediaReader::TDTFMediaReader ( )

Constructor.

6.96.3 Member Function Documentation

6.96.3.1 bool Coral::TDTFMediaReader::checkInterval (card64 & *time*, bool & *newRUList*)

Check, if new interval is reached.

Parameters

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

Returns

true, if new interval is reached.

6.96.3.2 void Coral::TDTFMediaReader::close ( ) [virtual]

Close media, if opened.

Reimplemented from Coral::TDTFReader.

6.96.3.3 MediaError Coral::TDTFMediaReader::getErrorCode ( ) const [inline]

Get error code.

Returns

Error code.

6.96.3.4 double Coral::TDTFMediaReader::getFrameRate ( ) const [inline]

Get frame rate.

Returns

Frame rate.

6.96.3.5 card64 Coral::TDTFMediaReader::getMaxPosition ( ) const [inline]

Get maximum position.

Returns

maximum position in nanoseconds.

6.96.3.6 void Coral::TDTFMediaReader::getMediaInfo (MediaInfo & *mediaInfo*) const

Get MediaInfo.

Parameters

mediaInfo	Reference to store media info.

6.96.3.7 cardinal Coral::TDTFMediaReader::getNextBlock (const cardinal *layers*, cardinal * *frameIDArray*, cardinal * *frameSizeArray*)

Read next frame trace block.

Parameters

layers	Maximum number of layers to store in arrays.
frameID-Array	Pointer to array to store frame-IDs.
frameSize-Array	Pointer to array to store frame sizes.
newInterval	Reference to boolean to store true, if new interval is reached; false otherwise.

Returns

cardinal Number of layers read.

6.96.3.8 card64 Coral::TDTFMediaReader::getPosition ( ) const [inline]

Get current position.

Returns

Position in nanoseconds.

6.96.3.9 bool Coral::TDTFMediaReader::open (const char * *name*, const bool *readWrite* = `false`, const bool *ruRequired* = `false`)

Open media.

Parameters

name	Name of media, e.g. a file name.
readWrite	true to open file in read/write mode; false for read-only.
ruRequired	true, if file has to contain resource/utilization list for successful opening; false otherwise.

Returns

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

6.96.3.10 bool Coral::TDTFMediaReader::ready ( ) const [inline]

Check, if TDTFMediaReader is ready for reading.

Returns

true, if TDTFMediaReader is ready; false otherwise.

6.96.3.11 double Coral::TDTFMediaReader::setFrameRate (const double *frameRate*) [inline]

Set frame rate.

Parameters

frameRate	Frame rate.

Returns

Frame rate set.

6.96.3.12 void Coral::TDTFMediaReader::setPosition (const card64 *position*) [inline]

Get position.

Parameters

position	Position in nanoseconds.

6.96.4 Member Data Documentation

6.96.4.1 MediaError Coral::TDTFMediaReader::ErrorCode [private]

6.96.4.2 double Coral::TDTFMediaReader::FrameRate [private]

6.96.4.3 const IntervalHeader* Coral::TDTFMediaReader::Interval [private]

6.96.4.4 card64 Coral::TDTFMediaReader::MaxPosition [private]

6.96.4.5 card64 Coral::TDTFMediaReader::Position [private]

6.96.4.6 const ResourceUtilizationHeader* Coral::TDTFMediaReader::RUHeader [private]

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

- tdtfmediareader.h
- tdtfmediareader.cc

6.97 Coral::TDTFPrefix Struct Reference

TDTF Prefix.

`#include <tdtf.h>`

Public Types

- enum MediaTypes { MT_Unknown = 0x00, MT_MPEG = 0x01, MT_H263 = 0x02, MT_MP3 = 0x03 }

Public Attributes

- char TDTF_ID [4]
- card32 Version
- card64 TraceTimeStamp
- card64 UtilizationTimeStamp
- card16 MediaType
- card16 MediaSubtype
- card32 pad01
- card64 pad02
- char Title [MaxTitleLength]
- char Copyright [MaxCopyrightLength]
- char Comment [MaxCommentLength]
- char URL [MaxURLLength]

Static Public Attributes

- static const card32 CurrentVersion = 0x74660000
- static const cardinal MaxTitleLength = 64
- static const cardinal MaxCopyrightLength = 64
- static const cardinal MaxCommentLength = 64
- static const cardinal MaxURLLength = 128

6.97.1 Detailed Description

TDTF Prefix.

This is a TDTF file's prefix.

Author

Thomas Dreibholz `dreibh@iem.uni-due.de`

Version

1.0

6.97.2 Member Enumeration Documentation

6.97.2.1 enum Coral::TDTFPrefix::MediaTypes

Media type constants.

Enumerator:

MT_Unknown

MT_MPEG

MT_H263

MT_MP3

6.97.3 Member Data Documentation

6.97.3.1 char Coral::TDTFPrefix::Comment[MaxCommentLength]

Comment.

6.97.3.2 char Coral::TDTFPrefix::Copyright[MaxCopyrightLength]

Copyright.

6.97.3.3 const card32 Coral::TDTFPrefix::CurrentVersion = 0x74660000 [static]

Current format version.

6.97.3.4 const cardinal Coral::TDTFPrefix::MaxCommentLength = 64 [static]

Maximum comment length.

6.97.3.5 const cardinal Coral::TDTFPrefix::MaxCopyrightLength = 64 [static]

Maximum copyright length.

6.97.3.6 const cardinal Coral::TDTFPrefix::MaxTitleLength = 64 [static]

Maximum title length.

6.97.3.7 const cardinal Coral::TDTFPrefix::MaxURLLength = 128 [static]

Maximum URL length.

6.97.3.8 card16 Coral::TDTFPrefix::MediaSubtype

Media subtype;

6.97.3.9 card16 Coral::TDTFPrefix::MediaType

Media type.

6.97.3.10 card32 Coral::TDTFPrefix::pad01

Unused.

6.97.3.11 card64 Coral::TDTFPrefix::pad02

Unused.

6.97.3.12 char Coral::TDTFPrefix::TDTF_ID[4]

This is the TDTF ID: "TDTF".

6.97.3.13 char Coral::TDTFPrefix::Title[MaxTitleLength]

Title.

6.97.3.14 card64 Coral::TDTFPrefix::TraceTimeStamp

Time stamp for start of trace creation. It can be used to compute total creation time using TDTFSuffix's TraceTimeStamp.

See also

TDTFSuffix::TraceTimeStamp

6.97.3.15 char Coral::TDTFPrefix::URL[MaxURLLength]

URL.

6.97.3.16 card64 Coral::TDTFPrefix::UtilizationTimeStamp

Time stamp for start of utilization creation. It can be used to compute total creation time using TDTFSuffix's UtilizationTimeStamp. If this time stamp is 0, a utilization has not been created yet or is invalid (e.g. TUtilizer abortion or failure). Therefore, it can also be used to check, if utilization information is valid.

See also

TDTFSuffix::UtilizationTimeStamp

6.97.3.17 card32 Coral::TDTFPrefix::Version

Format version.

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

- tdtf.h

6.98 Coral::TDTFPrefixExtensionHeader Struct Reference

TDTF Prefix Extension Header.

`#include <tdtf.h>`

Public Attributes

- char ExtID [4]
- card32 ExtLength

6.98.1 Detailed Description

TDTF Prefix Extension Header.

This is the TDTF prefix extension header.

Author

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

6.98.2 Member Data Documentation

6.98.2.1 char Coral::TDTFPrefixExtensionHeader::ExtID[4]

ID of this extension (0x00000000, Length 0 for no more extensions).

6.98.2.2 card32 Coral::TDTFPrefixExtensionHeader::ExtLength

Length of the extension.

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

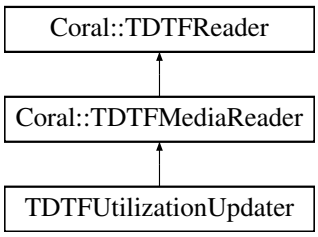
- tdtf.h

6.99 Coral::TDTFReader Class Reference

Trace Reader.

`#include <tdtfreader.h>`

Inheritance diagram for Coral::TDTFReader:



Classes

- struct MediaCacheEntry

Public Member Functions

- TDTFReader ()
- virtual ~TDTFReader ()
- virtual bool open (const char *name, const bool readWrite=false)
- virtual void close ()
- bool hasResourceUtilizationLists () const
- card16 getMediaType () const
- card16 getMediaSubtype () const
- bool isValidFrameRate (const double frameRate) const
- double getNearestValidFrameRate (const double frameRate) const
- double getMinFrameRate () const
- double getMaxFrameRate () const
- double getNextFrameRateForRate (const double frameRate) const
- double getPrevFrameRateForRate (const double frameRate) const
- cardinal getMaxByteCountForDelay (const card64 rtpPosition, const cardinal layer, const double frameRate, const cardinal delay) const

- cardinal getMaxFrameCountForDelay (const card64 rtpPosition, const cardinal layer, const double frameRate, const cardinal delay) const
- cardinal getPayloadFrameSizeForDelay (const card64 rtpPosition, const cardinal layer, const double frameRate, const cardinal bufferDelay) const
- cardinal getTraceFrameSize (const card64 rtpPosition, const cardinal layer, const double frameRate) const
- cardinal getTraceFrameID (const card64 rtpPosition, const cardinal layer, const double frameRate) const
- double getLayerScalability (const card64 rtpPosition, const cardinal layer, const double frameRate) const
- cardinal getLayerFlags (const card64 rtpPosition, const cardinal layer, const double frameRate) const
- cardinal getFrames (const double frameRate) const
- cardinal getLayers (const double frameRate) const
- cardinal getMaxBufferDelay (const card64 rtpPosition, const double frameRate) const
- bool checkAccess (const cardinal position, const cardinal length) const
- const TDTFPrefix * TDTFReader::getTDTFPrefix () const
- const TDTFSuffix * TDTFReader::getTDTFSuffix () const
- const MainIndexEntry * getMainIndexEntry (const double frameRate) const
- const TraceHeader * getTraceHeader (const double frameRate) const
- const PositionLengthIntervalIndexHeader * getIPLIHeader (const double frameRate) const
- const IntervalHeader * getIntervalHeader (const cardinal position, const double frameRate) const
- const LayerHeader * getLayerHeader (const cardinal position, const cardinal layer, const double frameRate) const
- const EmpiricalEnvelope * getEmpiricalEnvelope (const cardinal position, const cardinal layer, const double frameRate, const bool frameCount=false) const
- const ResourceUtilizationHeader * getResourceUtilizationHeader (const cardinal position, const double frameRate) const
- const UtilizationHeader * getFrameSizeUtilizationHeader (const cardinal position, const cardinal layer, const double frameRate) const
- const UtilizationHeader * getFrameRateUtilizationHeader (const cardinal position, const double frameRate) const
- cardinal positionToFramePosition (const double frameRate, const card64 rtpPosition) const
- card64 framePositionToPosition (const double frameRate, const cardinal framePosition) const
- void printEmpiricalEnvelope (ostream &os, const cardinal position, const double frameRate, const bool frameCount, const bool compact) const
- void print (ostream &os, const double frameRate, const bool printEE, const bool compactEE, const bool printRUL, const bool printUC) const

Public Attributes

- int InputFile
- char * InputMemory
- cardinal InputLength
- MediaCacheEntry * InputEntry

Protected Attributes

- MainIndexHeader * MainIndex
- cardinal MainIndexEntries
- ResourceUtilizationListIndexHeader * RULIndex
- cardinal RULIndexEntries
- double MinFrameRate
- double MaxFrameRate
- card16 MediaType
- card16 MediaSubtype

Static Private Attributes

- static multimap< const String, MediaCacheEntry * > MediaCache
- static Synchronizable MediaCacheSync

6.99.1 Detailed Description

Trace Reader.

This class is a reader for a TDTF trace file. The file will be mapped to memory using mmap() to provide easy access to trace and interval data.

Author

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

6.99.2 Constructor & Destructor Documentation

6.99.2.1 Coral::TDTFReader::TDTFReader ( )

Constructor.

6.99.2.2 Coral::TDTFReader::~TDTFReader ( ) [virtual]

Destructor.

6.99.3 Member Function Documentation

6.99.3.1 bool Coral::TDTFReader::checkAccess (const cardinal *position,* const cardinal *length*) const

Check, if access to trace file at given position for given length is within valid range.

Parameters

position	File position.
length	Access length.

Returns

true, is access is valid; false otherwise.

6.99.3.2 void Coral::TDTFReader::close ( ) [virtual]

Close TDTF trace file.

Reimplemented in Coral::TDTFMediaReader.

6.99.3.3 card64 Coral::TDTFReader::framePositionToPosition (const double *frameRate,* const cardinal *framePosition*) const [inline]

Convert frame position to RTP position.

Parameters

frameRate	Frame rate.
frame-Position	Frame RTP position.

Returns

RTP position.

6.99.3.4 const EmpiricalEnvelope * Coral::TDTFReader::getEmpiricalEnvelope (const cardinal *position,* const cardinal *layer,* const double *frameRate,* const bool *frameCount* = false) const

Get empirical envelope for byterate or frame count for given frame rate, layer and position.

Parameters

position	Position.
layer	Layer.
frameRate	Frame rate.
frameCount	true to get empirical envelope for frameCount; false for byterate.

Returns

Pointer to empirical envelope header.

6.99.3.5 const UtilizationHeader * Coral::TDTFReader::getFrameRate-UtilizationHeader (const cardinal *position*, const double *frameRate*) const

Get frame rate utilization constants for given frame rate, layer and position.

Parameters

position	Position.
frameRate	Frame rate.
a	Reference to store constant A.
b	Reference to store constant B.
c	Reference to store constant C.

Returns

true on success; false otherwise.
UtilizationHeader.

6.99.3.6 cardinal Coral::TDTFReader::getFrames (const double *frameRate*) const

Get number of frames for given frame rate.

Parameters

frameRate	Frame rate.

Returns

Number of frames.

6.99.3.7 const UtilizationHeader * Coral::TDTFReader::getFrameSizeUtilizationHeader (const cardinal *position,* const cardinal *layer,* const double *frameRate*) const

Get frame size utilization constants for given frame rate, layer and position.

Parameters

position	Position.
layer	Layer.
frameRate	Frame rate.

Returns

UtilizationHeader.

6.99.3.8 const IntervalHeader * Coral::TDTFReader::getIntervalHeader (const cardinal *position,* const double *frameRate*) const

Get interval header for given frame rate and position.

Parameters

position	Position.
frameRate	Frame rate.

Returns

Pointer to trace header.

6.99.3.9 const PositionLengthIntervalIndexHeader * Coral::TDTFReader::getIPLIHeader (const double *frameRate*) const

Get entry for given frame rate from Position/Length/Interval index.

Parameters

frameRate	Frame rate.

Returns

Pointer to entry.

6.99.3.10 cardinal Coral::TDTFReader::getLayerFlags (const card64 *rtpPosition,* const cardinal *layer,* const double *frameRate*) const

Get layer flags.

Parameters

rtpPosition	RTP position within media.
layer	Layer number.
frameRate	Frame rate.

Returns

Layer flags.

6.99.3.11 const LayerHeader * Coral::TDTFReader::getLayerHeader (const cardinal *position,* const cardinal *layer,* const double *frameRate*) const

Get layer header for given frame rate, position and layer.

Parameters

position	Position.
layer	Layer.
frameRate	Frame rate.

Returns

Pointer to layer header.

6.99.3.12 cardinal Coral::TDTFReader::getLayers (const double *frameRate*) const

Get number of layers for given frame rate.

Parameters

frameRate	Frame rate.

Returns

Number of layers.

6.99.3.13 double Coral::TDTFReader::getLayerScalability (const card64 *rtpPosition,* const cardinal *layer,* const double *frameRate*) const

Get layer scalability.

Parameters

rtpPosition	RTP position within media.
layer	Layer number.
frameRate	Frame rate.

Returns

Layer scalability.

6.99.3.14 const MainIndexEntry * Coral::TDTFReader::getMainIndexEntry (const double *frameRate*) const

Get entry for given frame rate from main index.

Parameters

frameRate	Frame rate.

Returns

Pointer to entry.

6.99.3.15 cardinal Coral::TDTFReader::getMaxBufferDelay (const card64 *rtpPosition,* const double *frameRate*) const

Get maximum buffer delay for given position and frame rate.

Parameters

rtpPosition	RTP position within media.
frameRate	Frame rate.

Returns

Maximum buffer delay

6.99.3.16 cardinal Coral::TDTFReader::getMaxByteCountForDelay (const card64 *rtpPosition,* const cardinal *layer,* const double *frameRate,* const cardinal *delay*) const

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

Parameters

rtpPosition	RTP position within media.
layer	Layer number.
frameRate	Frame rate.
bufferDelay	Buffer delay in frame rate units.

Returns

Maximum number of bytes.

6.99.3.17 cardinal Coral::TDTFReader::getMaxFrameCountForDelay (const card64 *rtpPosition,* const cardinal *layer,* const double *frameRate,* const cardinal *delay*) const

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

Parameters

rtpPosition	RTP position within media.
layer	Layer number.
frameRate	Frame rate.
bufferDelay	Buffer delay in frame rate units.

Returns

Maximum number of frames.

6.99.3.18 double Coral::TDTFReader::getMaxFrameRate ( ) const [inline]

Get maximum frame rate.

Returns

Maximum frame rate.

6.99.3.19 card16 Coral::TDTFReader::getMediaSubtype ( ) const [inline]

Get media subtype.

Returns

Media subtype.

6.99.3.20 card16 Coral::TDTFReader::getMediaType ( ) const [inline]

Get media type.

Returns

Media type.

See also

TDTFPrefix::MediaTypes

6.99.3.21 double Coral::TDTFReader::getMinFrameRate ( ) const [inline]

Get minimum frame rate.

Returns

Minimum frame rate.

6.99.3.22 double Coral::TDTFReader::getNearestValidFrameRate (const double *frameRate*) const

Get nearest lower valid frame rate for given frame rate.

Parameters

	rate	Frame rate.

Returns

Valid frame rate nearest to given rate.

6.99.3.23 double Coral::TDTFReader::getNextFrameRateForRate (const double *frameRate*) const

Get next higher valid frame rate for given frame rate.

Parameters

frameRate	Frame rate.

Returns

Next higher valid frame rate.

6.99.3.24 cardinal Coral::TDTFReader::getPayloadFrameSizeForDelay (const card64 *rtpPosition,* const cardinal *layer,* const double *frameRate,* const cardinal *bufferDelay*) const [inline]

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

Parameters

rtpPosition	RTP position within media.
layer	Layer number.
frameRate	Frame rate.
bufferDelay	Buffer delay in frame rate units.

Returns

Payload frame size.

6.99.3.25 double Coral::TDTFReader::getPrevFrameRateForRate (const double *frameRate*) const

Get next lower valid frame rate for given frame rate.

Parameters

frameRate	Frame rate.

Returns

Next lower valid frame rate.

6.99.3.26 const ResourceUtilizationHeader * Coral::TDTFReader::getResourceUtilizationHeader (const cardinal *position,* const double *frameRate*) const

Get resource/utilization header for given position and frame rate.

Parameters

position	Position refering to ∗maximum∗ frame rate.
frameRate	Frame rate.

Returns

ResourceUtilizationHeader.

6.99.3.27 cardinal Coral::TDTFReader::getTraceFrameID (const card64 *rtpPosition,* const cardinal *layer,* const double *frameRate*) const

Get frame ID for given position, layer and frame rate from trace.

Parameters

rtpPosition	RTP position within media.
layer	Layer number.
frameRate	Frame rate.

Returns

Frame ID.

6.99.3.28 cardinal Coral::TDTFReader::getTraceFrameSize (const card64 *rtpPosition,* const cardinal *layer,* const double *frameRate*) const

Get payload frame size for given position, layer and frame rate from trace.

Parameters

rtpPosition	RTP position within media.
layer	Layer number.
frameRate	Frame rate.

Returns

Frame size.

6.99.3.29 const TraceHeader ∗ Coral::TDTFReader::getTraceHeader (const double *frameRate*) const

Get trace header for given frame rate.

Parameters

frameRate	Frame rate.

Returns

Pointer to trace header.

6.99.3.30 bool Coral::TDTFReader::hasResourceUtilizationLists ( ) const [inline]

Check, if TDTF file has got resource/utilization lists.

Returns

true, if lists are included; false otherwise.

6.99.3.31 bool Coral::TDTFReader::isValidFrameRate (const double *frameRate*) const

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.

6.99.3.32 bool Coral::TDTFReader::open (const char * *name,* const bool *readWrite* = false) [virtual]

Open TDTF trace file.

Parameters

name	Name of TDTF trace file to open.
writable	true, to open file in read/write mode; false otherwise.

Returns

true, if open operation has been successful; false otherwise.

6.99.3.33 cardinal Coral::TDTFReader::positionToFramePosition (const double *frameRate,* const card64 *rtpPosition*) const [inline]

Convert RTP position to frame position.

Parameters

frameRate	Frame rate.
rtpPosition	RTP position.

Returns

Frame position.

6.99.3.34 void Coral::TDTFReader::print (ostream & *os,* const double *frameRate,* const bool *printEE,* const bool *compactEE,* const bool *printRUL,* const bool *printUC*) const

Print complete TDTF trace for given frame rate.

Parameters

os	Output stream.
frameRate	Frame rate.
printEE	true, to print empirical envelope; false otherwise.
compactEE	true, to print Empirical envelope ∗without∗ approximations; false otherwise.
printRUL	true, to print resource/utilization list; false otherwise.
printUC	true, to print utilization constants; false otherwise.

6.99.3.35 void Coral::TDTFReader::printEmpiricalEnvelope (ostream & *os,* const cardinal *position,* const double *frameRate,* const bool *frameCount,* const bool *compact*) const

Print empirical envelopes for all layers for byterate or frame count for given frame rate and position.

Parameters

os	Output stream.
frameRate	Frame rate.
position	Position.
compact	true, to print ∗without∗ EE approximations; false otherwise.
frameCount	true to get empirical envelope for frameCount; false for byterate.

6.99.3.36 const TDTFPrefix* Coral::TDTFReader::TDTFReader::getTDTFPrefix ( ) const

Get TDTF prefix.

Returns

Pointer to TDTF prefix.

6.99.3.37 const TDTFSuffix* Coral::TDTFReader::TDTFReader::getTDTFSuffix ( ) const

Get TDTF suffix.

Returns

Pointer to TDTF suffix.

6.99.4 Member Data Documentation

6.99.4.1 MediaCacheEntry* Coral::TDTFReader::InputEntry

6.99.4.2 int Coral::TDTFReader::InputFile

6.99.4.3 cardinal Coral::TDTFReader::InputLength

6.99.4.4 char* Coral::TDTFReader::InputMemory

6.99.4.5 MainIndexHeader* Coral::TDTFReader::MainIndex [protected]

6.99.4.6 cardinal Coral::TDTFReader::MainIndexEntries [protected]

6.99.4.7 double Coral::TDTFReader::MaxFrameRate [protected]

6.99.4.8 multimap< const String, TDTFReader::MediaCacheEntry * > Coral::TDTFReader::MediaCache [static, private]

6.99.4.9 Synchronizable Coral::TDTFReader::MediaCacheSync [static, private]

6.99.4.10 card16 Coral::TDTFReader::MediaSubtype [protected]

6.99.4.11 card16 Coral::TDTFReader::MediaType [protected]

6.99.4.12 double Coral::TDTFReader::MinFrameRate [protected]

6.99.4.13 ResourceUtilizationListIndexHeader* Coral::TDTFReader::RULIndex [protected]

6.99.4.14 cardinal Coral::TDTFReader::RULIndexEntries [protected]

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

- tdtfreader.h
- tdtfreader.cc

6.100 Coral::TDTFSuffix Struct Reference

TDTF Suffix.

`#include <tdtf.h>`

Public Attributes

- card64 TraceTimeStamp
- card64 UtilizationTimeStamp
- card32 ResourceUtilizationStart
- card32 ResourceUtilizationIndex
- card32 MainIndex
- card32 TraceStart
- char TDTF_ID [4]

6.100.1 Detailed Description

TDTF Suffix.

This is a TDTF file's suffix.

Author

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

6.100.2 Member Data Documentation

6.100.2.1 card32 Coral::TDTFSuffix::MainIndex

File position of the main index.

6.100.2.2 card32 Coral::TDTFSuffix::ResourceUtilizationIndex

Resource/utilization index;

6.100.2.3 card32 Coral::TDTFSuffix::ResourceUtilizationStart

Resource/utilization block start file position.

6.100.2.4 char Coral::TDTFSuffix::TDTF_ID[4]

This is the TDTF ID: "TDTF".

6.100.2.5 card32 Coral::TDTFSuffix::TraceStart

Trace block start file position.

6.100.2.6 card64 Coral::TDTFSuffix::TraceTimeStamp

Time stamp for end of trace creation. It can be used to compute total creation time using TDTFPrefix's TraceTimeStamp.

See also

TDTFPrefix::TraceTimeStamp

6.100.2.7 card64 Coral::TDTFSuffix::UtilizationTimeStamp

Time stamp for end of utilization creation. It can be used to compute total creation time using TDTFPrefix's TraceTimeStamp.

See also

TDTFPrefix::UtilizationTimeStamp

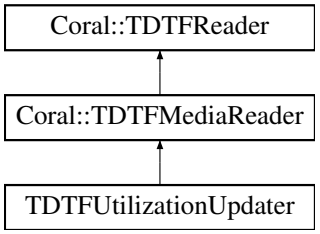
The documentation for this struct was generated from the following file:

- tdtf.h

6.101 TDTFUtilizationUpdater Class Reference

TDTF Media Reader.

Inheritance diagram for TDTFUtilizationUpdater:



Public Member Functions

- bool doUpdate (TraceQoSDescription *qosDescription, const TraceConfiguration &config, const bool truncateOnly)

6.101.1 Detailed Description

TDTF Media Reader.

This class extends the TDTF media reader by a method to do utilization calculations.

Author

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

6.101.2 Member Function Documentation

6.101.2.1 bool TDTFUtilizationUpdater::doUpdate (TraceQoSDescription * *qosDescription,* const TraceConfiguration & *config,* const bool *truncateOnly*)

Write new utilization information to TDTF file represented by TDTFMediaReader. After the calculation, the file will be closed!

Parameters

qos-Description	QoS description for the trace's media type.
config	TraceConfiguration.

Returns

true, if operation was successful; false otherwise.

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

- tutilizer.cc

6.102 Coral::TDTFWriter Class Reference

TDTF Writer.

`#include <tdtfwriter.h>`

Public Member Functions

- TDTFWriter ()
- ~TDTFWriter ()
- void generate (const char *name, TraceArray *originalTraceArray, const TraceConfiguration &config)

Private Member Functions

- cardinal writeTrace ()
- cardinal writeIntervals ()

Private Attributes

- FILE * OutputFile
- TraceArray * TArray
- TraceConfiguration Config

6.102.1 Detailed Description

TDTF Writer.

This class is a writer for TDTF files.

Author

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

6.102.2 Constructor & Destructor Documentation

6.102.2.1 Coral::TDTFWriter::TDTFWriter ( )

Constructor.

6.102.2.2 Coral::TDTFWriter::~TDTFWriter ( )

Destructor.

6.102.3 Member Function Documentation

6.102.3.1 void Coral::TDTFWriter::generate (const char * *name*, TraceArray * *originalTraceArray*, const TraceConfiguration & *config*)

Generator TDTF file.

Parameters

name	File name for TDTF file to generate.
original-TraceArray	TraceArray containing input trace.
config	Configuration for the TDTF trace generator.

6.102.3.2 cardinal Coral::TDTFWriter::writeIntervals ( ) [private]

6.102.3.3 cardinal Coral::TDTFWriter::writeTrace ( ) [private]

6.102.4 Member Data Documentation

6.102.4.1 TraceConfiguration Coral::TDTFWriter::Config [private]

6.102.4.2 FILE* Coral::TDTFWriter::OutputFile [private]

6.102.4.3 TraceArray* Coral::TDTFWriter::TArray [private]

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

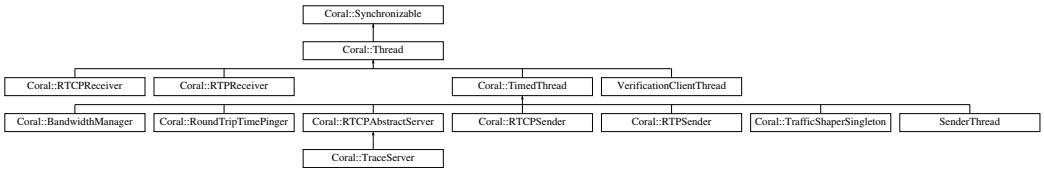
- tdtfwriter.h
- tdtfwriter.cc

6.103 Coral::Thread Class Reference

Thread.

`#include <thread.h>`

Inheritance diagram for Coral::Thread:



Public Member Functions

- Thread (const cardinal flags=ThreadCancelAsynchronous)
- virtual ~Thread ()
- bool running () const
- virtual bool start ()
- virtual void stop ()
- cardinal join ()
- virtual void cancel ()
- virtual void suspend ()
- virtual void resume ()

Static Public Member Functions

- static card64 delay (const card64 delayTimeout, const bool interruptable=false)

Static Public Attributes

- static const cardinal ThreadCancelAsynchronous = 0
- static const cardinal ThreadCancelDeferred = (1 << 0)

Protected Member Functions

- virtual void testCancel ()
- virtual bool testSuspension ()
- virtual void run ()=0

Static Protected Member Functions

- static void exit (const cardinal result=0)
- static void yield ()

Protected Attributes

- pthread_t PThread

Static Private Member Functions

- static void * go (void *i)

Private Attributes

- pthread_mutex_t SuspensionMutex
- cardinal Flags
- bool IsSuspended

6.103.1 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

6.103.2 Constructor & Destructor Documentation

6.103.2.1 Coral::Thread::Thread (const cardinal *flags* = ThreadCancelAsynchronous)

Constructor. A new thread will be created but *not* started! To start the new thread, call start().

Parameters

flags	Flags for the thread to be created

See also

start

6.103.2.2 Coral::Thread::~Thread ( ) [virtual]

Destructor. The thread will be stopped (if running) and deleted.

6.103.3 Member Function Documentation

6.103.3.1 void Coral::Thread::cancel ( ) [virtual]

Cancel the thread.

6.103.3.2 card64 Coral::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.

6.103.3.3 static void Coral::Thread::exit (const cardinal *result* = 0) [inline, static, protected]

Exit current thread.

Parameters

result	Result to return.

6.103.3.4 void * Coral::Thread::go (void * *i*) [static, private]

6.103.3.5 cardinal Coral::Thread::join ( )

Wait for the thread to be finished.

6.103.3.6 void Coral::Thread::resume ( ) [virtual]

Resume a suspended thread.

6.103.3.7 virtual void Coral::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 Coral::RTPReceiver, Coral::TimedThread, Coral::RTCPReceiver, and VerificationClientThread.

6.103.3.8 bool Coral::Thread::running ( ) const [inline]

Check, if the thread is running.

Returns

true, if the thread is running, false otherwise

6.103.3.9 bool Coral::Thread::start ( ) [virtual]

Start the thread, if not already started.

Returns

true, if the thread has been started; false, if not.

6.103.3.10 void Coral::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

Reimplemented in Coral::RTCPAbstractServer, Coral::TimedThread, and VerificationClientThread.

6.103.3.11 void Coral::Thread::suspend ( ) [virtual]

Suspend the thread. Note: The thread will not be suspended immediately! The thread will be suspended, when it reaches the next testSuspension() call! => To implement suspendable threads, testSuspension() must be called by the thread regularly!!!

See also

testSuspension

6.103.3.12 void Coral::Thread::testCancel ( ) [protected, virtual]

Test for cancellation. If the thread received a cancel signal, it will be cancelled.

6.103.3.13 bool Coral::Thread::testSuspension ( ) [protected, virtual]

Test for suspension. If the thread received a suspension signal, it will be suspended.

Returns

true, if the thread was suspended.

6.103.3.14 static void Coral::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.

6.103.4 Member Data Documentation

6.103.4.1 cardinal Coral::Thread::Flags [private]

6.103.4.2 bool Coral::Thread::IsSuspended [private]

6.103.4.3 pthread_t Coral::Thread::PThread [protected]

6.103.4.4 pthread_mutex_t Coral::Thread::SuspensionMutex [private]

6.103.4.5 const cardinal Coral::Thread::ThreadCancelAsynchronous = 0 [static]

6.103.4.6 const cardinal Coral::Thread::ThreadCancelDeferred = (1 << 0) [static]

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

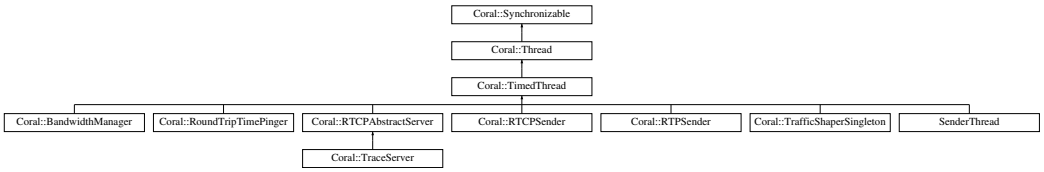
- thread.h
- thread.cc

6.104 Coral::TimedThread Class Reference

Timed Thread.

`#include <timedthread.h>`

Inheritance diagram for Coral::TimedThread:



Public Member Functions

- TimedThread (const card64 usec, const cardinal flags=ThreadCancelAsynchronous)
- ~TimedThread ()
- card64 getInterval () const
- void setInterval (const card64 usec)
- cardinal getTimerCorrection () const
- void setTimerCorrection (const cardinal maxCorrection=0)
- void leaveCorrectionLoop ()
- void setFastStart (const bool on)
- bool getFastStart () const
- void stop ()
- bool pendingTimerEvent () const

Protected Member Functions

- virtual void timerEvent ()=0

Private Member Functions

- void run ()

Private Attributes

- card64 Interval
- cardinal TimerCorrection
- bool AlarmHandlerInitialized
- bool FastStart
- bool NewInterval
- bool LeaveCorrectionLoop

6.104.1 Detailed Description

Timed Thread.

This abstract class realizes a timed thread 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). 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

Thread

6.104.2 Constructor & Destructor Documentation

6.104.2.1 Coral::TimedThread::TimedThread (const card64 *usec,* const cardinal *flags* = ThreadCancelAsynchronous)

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.
flags	Thread flags

See also

Thread::start
timerEvent
Thread::Thread
setTimerCorrection
setFastStart

6.104.2.2 Coral::TimedThread::~TimedThread ( )

Destructor.

6.104.3 Member Function Documentation

6.104.3.1 bool Coral::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.

6.104.3.2 card64 Coral::TimedThread::getInterval ( ) const [inline]

Get timed thread's interval.

Returns

Interval in microseconds.

6.104.3.3 cardinal Coral::TimedThread::getTimerCorrection ( ) const [inline]

Get maxCorrection value for inaccurate system timer.

Returns

true, if activated; false if not.

See also

setTimerCorrection

6.104.3.4 void Coral::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.

6.104.3.5 bool Coral::TimedThread::pendingTimerEvent ( ) const [inline]

Check for pending timer event(). This can be used to check for a pending timer event() (SIGALRM signal) within the current timerEvent() run.

6.104.3.6 void Coral::TimedThread::run ( ) [private, virtual]

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

Implements Coral::Thread.

6.104.3.7 void Coral::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.

6.104.3.8 void Coral::TimedThread::setInterval (const card64 *usec*)

Set timed thread's interval.

Parameters

usec	Interval in microseconds.

6.104.3.9 void Coral::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.

6.104.3.10 void Coral::TimedThread::stop ( ) [virtual]

Reimplementation of stop().

Reimplemented from Coral::Thread.

Reimplemented in Coral::RTCPAbstractServer.

6.104.3.11 virtual void Coral::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 Coral::BandwidthManager, Coral::RTPSender, Coral::RTCPAbstractServer, Coral::RoundTripTimePinger, Coral::RTCPSender, Coral::TrafficShaperSingleton, and SenderThread.

6.104.4 Member Data Documentation

6.104.4.1 bool Coral::TimedThread::AlarmHandlerInitialized [private]

6.104.4.2 bool Coral::TimedThread::FastStart [private]

6.104.4.3 card64 Coral::TimedThread::Interval [private]

6.104.4.4 bool Coral::TimedThread::LeaveCorrectionLoop [private]

6.104.4.5 bool Coral::TimedThread::NewInterval [private]

6.104.4.6 cardinal Coral::TimedThread::TimerCorrection [private]

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

- timedthread.h
- timedthread.cc

6.105 Coral::TraceArray::Trace Struct Reference

`#include <tracearray.h>`

Public Attributes

- cardinal Frames
- FrameDescription Frame [0]

6.105.1 Member Data Documentation

6.105.1.1 FrameDescription Coral::TraceArray::Trace::Frame[0]

6.105.1.2 cardinal Coral::TraceArray::Trace::Frames

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

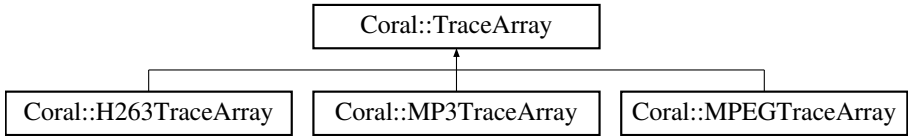
- tracearray.h

6.106 Coral::TraceArray Class Reference

Trace Array.

`#include <tracearray.h>`

Inheritance diagram for Coral::TraceArray:



Classes

- struct Trace

Public Member Functions

- TraceArray (const TraceConfiguration &config)
- virtual ~TraceArray ()
- void init (const cardinal maxLayers, const cardinal maxFrames)
- void calculateEmpiricalEnvelope (const cardinal layer, const cardinal start, const cardinal end, const cardinal traceStart, const cardinal traceEnd, EmpiricalEnvelope *ee, const bool frameCount=false) const
- void calculateOptimalIntervals (double *costArray, cardinal *lengthArray, const cardinal minLength, const cardinal maxLength) const
- double TraceArray::calculateTrafficCost (EmpiricalEnvelope **ee, const cardinal position, const cardinal length) const
- double TraceArray::calculateNextTrafficCost (EmpiricalEnvelope **ee, const cardinal position, const cardinal length) const
- virtual TraceArray * decreaseFrameRate () const
- EmpiricalEnvelope * TraceArray::initEmpiricalEnvelope (char *buffer, const cardinal eePairs, const bool frameCount=false) const

Public Attributes

- cardinal Layers
- cardinal Frames
- cardinal MaxLayers
- cardinal MaxFrames
- double FrameRate
- struct Coral::TraceArray::Trace ** LayerTrace

Protected Attributes

- TraceConfiguration Config

Private Member Functions

- cardinal calculateEmpiricalEnvelopePoint (const cardinal layer, const cardinal start, const cardinal end, const cardinal traceStart, const cardinal traceEnd, const cardinal delay, const bool frameCount=false) const

6.106.1 Detailed Description

Trace Array.

This is an array of layer traces.

Author

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

6.106.2 Constructor & Destructor Documentation

6.106.2.1 Coral::TraceArray::TraceArray (const TraceConfiguration & *config*)

Constructor.

Parameters

config	TraceConfiguration object with configuration of the trace.

6.106.2.2 Coral::TraceArray::~TraceArray ( ) [virtual]

Destructor.

6.106.3 Member Function Documentation

6.106.3.1 void Coral::TraceArray::calculateEmpiricalEnvelope (const cardinal *layer,* const cardinal *start,* const cardinal *end,* const cardinal *traceStart,* const cardinal *traceEnd,* EmpiricalEnvelope * *ee,* const bool *frameCount* = false) const

Calculate empirical envelope.

Parameters

layer	Layer number to calculate empirical envelope for.
start	Start position.
end	End position.
traceStart	Trace start position.
traceEnd	Trace end position.
ee	Initialized EmpiricalEnvelope to write sums into.
frameCount	true, to calculate empirical envelope for frame count; false for byterate.

6.106.3.2 cardinal Coral::TraceArray::calculateEmpiricalEnvelopePoint (const cardinal *layer,* const cardinal *start,* const cardinal *end,* const cardinal *traceStart,* const cardinal *traceEnd,* const cardinal *delay,* const bool *frameCount* = `false`) const `[private]`

6.106.3.3 void Coral::TraceArray::calculateOptimalIntervals (double * *costArray,* cardinal * *lengthArray,* const cardinal *minLength,* const cardinal *maxLength*) const

Calculate optimal intervals refering to given configuration.

Parameters

costArray	Array to store costs into.
lengthArray	Array to store lengths into.
minLength	Minimum length.
maxLength	Maximum length.

6.106.3.4 TraceArray * Coral::TraceArray::decreaseFrameRate ( ) const `[virtual]`

Decrease frame rate to next lower setting.

Returns

TraceArray containing next lower frame rate setting or NULL, if current setting is lowest.

6.106.3.5 void Coral::TraceArray::init (const cardinal *maxLayers,* const cardinal *maxFrames*)

Initialize trace array.

Parameters

maxLayers	Maximum number of layers.
maxFrames	Maximum number of frames.

6.106.3.6 double Coral::TraceArray::TraceArray::calculateNextTrafficCost (EmpiricalEnvelope ** *ee*, const cardinal *position*, const cardinal *length*) const [inline]

Update traffic cost calculation done by calculateTrafficCost() by an additional frame. In this case, the already calculated empirical envelope can be reused resulting in a huge speed improvement.

Parameters

ee	EmpiricalEnvelope array.
position	Position.
length	Length.

Returns

Cost.

6.106.3.7 double Coral::TraceArray::TraceArray::calculateTrafficCost (EmpiricalEnvelope ** *ee*, const cardinal *position*, const cardinal *length*) const

Calculate traffic cost for an interval.

Parameters

ee	EmpiricalEnvelope array.
position	Position.
length	Length.

Returns

Cost.

6.106.3.8 EmpiricalEnvelope* Coral::TraceArray::TraceArray::initEmpiricalEnvelope (char * *buffer*, const cardinal *eePairs*, const bool *frameCount* = false) const

Initialize EmpiricalEnvelope.

Parameters

buffer	Buffer for empirical envelope.
eePairs	Number of D-BIND pairs.
frameCount	true, if empirical envelope is for frame count; false for byterate.

Returns

Required size.

6.106.4 Member Data Documentation

6.106.4.1 TraceConfiguration Coral::TraceArray::Config [protected]

6.106.4.2 double Coral::TraceArray::FrameRate

6.106.4.3 cardinal Coral::TraceArray::Frames

6.106.4.4 cardinal Coral::TraceArray::Layers

6.106.4.5 struct Coral::TraceArray::Trace** Coral::TraceArray::LayerTrace

6.106.4.6 cardinal Coral::TraceArray::MaxFrames

6.106.4.7 cardinal Coral::TraceArray::MaxLayers

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

- tracearray.h
- tracearray.cc

6.107 Coral::TraceClient Class Reference

Trace Client.

`#include <traceclient.h>`

Public Member Functions

- TraceClient (const char *receiverName)
- ~TraceClient ()
- bool play (const char *server, const char *mediaName, const card32 sessionDescriptor=0)
- void change (const char *mediaName)
- void stop ()
- card64 getPosition ()
- card64 getMaxPosition () const
- MediaInfo getMediaInfo () const
- card8 getErrorCode () const
- const char * getEncoding () const
- card64 getMinWantedBandwidth () const
- card64 getMaxWantedBandwidth () const

- double getMaxTransferDelay () const
- double getWantedUtilization () const
- int8 getStreamPriority () const
- int8 getSessionPriority () const
- Range< card64 > getBandwidth () const
- double getFrameRate () const
- double getUtilization () const
- card32 getFlags () const
- cardinal getLayers () const
- card8 getIPVersion () const
- bool playing () const
- String getServerAddressString (InternetAddress::PrintFormat format=InternetAddress::PF_Address) const
- String getOurAddressString (InternetAddress::PrintFormat format=InternetAddress::PF_Address) 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)
- void setWantedUtilization (const double utilization)
- void setFlags (const card32 priority)
- void setMinWantedBandwidth (const card64 bandwidth)
- void setMaxWantedBandwidth (const card64 bandwidth)
- void setMaxTransferDelay (const double delay)
- void setStreamPriority (const int8 priority)
- void setSessionPriority (const int8 priority)
- void setEncoding (const cardinal index)

Private Member Functions

- void sendCommand (const bool updateRestartPosition=true)

Private Attributes

- RTPReceiver * Receiver
- RTCPSender * Sender
- Socket SenderSocket
- Socket ReceiverSocket
- InternetFlow Flow
- InternetAddress ServerAddress
- InternetAddress OurAddress
- card32 OurSSRC
- InternetAddress ReceiverAddress
- multimap< const cardinal, TraceDecoderInterface * > DecoderSet
- TraceDecoderRepository Decoders
- TraceClientAppPacket Status
- card64 OldPosition
- card64 ChangeTimeStamp
- bool IsPlaying

Static Private Attributes

- static const card64 RestartPositionUpdateDelay = 5000000

6.107.1 Detailed Description

Trace Client.

This class is a trace client.

Author

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

6.107.2 Constructor & Destructor Documentation

6.107.2.1 Coral::TraceClient::TraceClient (const char * *receiverName*)

Constructor for a new trace client.

Parameters

receiver-Name	String with the receiver name or NULL for default.

6.107.2.2 Coral::TraceClient::~TraceClient ( )

Destructor.

6.107.3 Member Function Documentation

6.107.3.1 void Coral::TraceClient::change (const char * *mediaName*)

Change media of an established connection.

Parameters

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

See also

play

6.107.3.2 Range<card64> Coral::TraceClient::getBandwidth ( ) const [inline]

Get bandwidth range of last transmission.

Returns

Bandwidth range.

6.107.3.3 card64 Coral::TraceClient::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

6.107.3.4 const char* Coral::TraceClient::getEncoding ( ) const [inline]

Get encoding name.

Returns

Encoding name.

6.107.3.5 const char * Coral::TraceClient::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.

6.107.3.6 card8 Coral::TraceClient::getErrorCode ( ) const [inline]

Get error code.

Returns

Error code.

6.107.3.7 card32 Coral::TraceClient::getFlags ( ) const [inline]

Get flags of last transmission.

Returns

Flags.

6.107.3.8 card32 Coral::TraceClient::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

6.107.3.9 double Coral::TraceClient::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

6.107.3.10 double Coral::TraceClient::getFrameRate ( ) const [inline]

Get frame rate of last transmission.

Returns

Frame rate.

6.107.3.11 InternetFlow Coral::TraceClient::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

6.107.3.12 card8 Coral::TraceClient::getIPVersion ( ) const

Get IP version.

Returns

IP Version.

6.107.3.13 double Coral::TraceClient::getJitter (const cardinal *layer* = 0) const

Get jitter for given layer.

Parameters

layer	Layer number.

Returns

Jitter.

See also

getLayers

6.107.3.14 cardinal Coral::TraceClient::getLayers ( ) const [inline]

Get number of layers of last transmission.

6.107.3.15 card64 Coral::TraceClient::getMaxPosition ( ) const [inline]

Get maximum media position.

Returns

Maximum position in nanoseconds.

6.107.3.16 double Coral::TraceClient::getMaxTransferDelay ( ) const [inline]

Get maximum transfer delay.

Returns

Maximum transfer delay.

6.107.3.17 card64 Coral::TraceClient::getMaxWantedBandwidth ( ) const [inline]

Get maximum wanted bandwidth.

Returns

Maximum wanted bandwidth.

6.107.3.18 MediaInfo Coral::TraceClient::getMediaInfo ( ) const

Get MediaInfo.

Returns

MediaInfo.

6.107.3.19 card64 Coral::TraceClient::getMinWantedBandwidth ( ) const [inline]

Get minimum wanted bandwidth.

Returns

Minimum wanted bandwidth.

6.107.3.20 String Coral::TraceClient::getOurAddressString (InternetAddress::PrintFormat *format* = InternetAddress::PF_Address) const

Get client address string.

Parameters

format	Print format.

Returns

Client address.

See also

InternetAddress::PrintFormat

6.107.3.21 card32 Coral::TraceClient::getOurSSRC ( ) const [inline]

Get client SSRC.

Returns

Client SSRC.

6.107.3.22 card64 Coral::TraceClient::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

6.107.3.23 card64 Coral::TraceClient::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

6.107.3.24 card64 Coral::TraceClient::getPosition ( )

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

Returns

Position in nanoseconds.

6.107.3.25 String Coral::TraceClient::getServerAddressString (InternetAddress::PrintFormat *format* = InternetAddress::PF_Address) const

Get server address string.

Parameters

format	Print format.

Returns

Server address.

See also

InternetAddress::PrintFormat

6.107.3.26 card32 Coral::TraceClient::getServerSSRC (const cardinal *layer* = 0) const

Get server SSRC for given layer.

Parameters

layer	Layer number.

Returns

Server SSRC.

See also

getLayers

6.107.3.27 int8 Coral::TraceClient::getSessionPriority ( ) const [inline]

Get session priority.

Returns

Session priority;

6.107.3.28 int8 Coral::TraceClient::getStreamPriority ( ) const [inline]

Get stream priority.

Returns

Stream priority;

6.107.3.29 card8 Coral::TraceClient::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

6.107.3.30 double Coral::TraceClient::getUtilization ( ) const [inline]

Get utilization of last transmission.

Returns

Utilization.

6.107.3.31 double Coral::TraceClient::getWantedUtilization ( ) const [inline]

Get wanted utilization.

Returns

Wanted utilization.

6.107.3.32 bool Coral::TraceClient::play (const char * *server*, const char * *mediaName*, const card32 *sessionDescriptor* = 0)

Start playing given media from given server.

Parameters

server	Server address (e.g. gaffel:7500).
mediaName	Media name (e.g. ../TraceFiles/Test1.list)
session-Descriptor	Session descriptor.

Returns

true, if play request has been sent to server.

6.107.3.33 bool Coral::TraceClient::playing ( ) const [inline]

Check, if trace client is playing.

Returns

true, if client is playing; false otherwise.

6.107.3.34 void Coral::TraceClient::sendCommand (const bool *updateRestartPosition* = true) [private]

6.107.3.35 void Coral::TraceClient::setEncoding (const cardinal *index*)

Set encoding by index in client's decoder repository.

Parameters

index	Index in decoder repository.

6.107.3.36 void Coral::TraceClient::setFlags (const card32 *priority*) [inline]

Set flags.

Parameters

flags	Flags.

6.107.3.37 void Coral::TraceClient::setMaxTransferDelay (const double *delay*) [inline]

Set maximum transfer delay.

Parameters

delay	Maximum transfer delay in microseconds.

6.107.3.38 void Coral::TraceClient::setMaxWantedBandwidth (const card64 *bandwidth*) [inline]

Set maximum wanted bandwidth.

Parameters

bandwidth	Maximum wanted bandwidth.

6.107.3.39 void Coral::TraceClient::setMinWantedBandwidth (const card64 *bandwidth*) [inline]

Set minimum wanted bandwidth.

Parameters

bandwidth	Minimum wanted bandwidth.

6.107.3.40 void Coral::TraceClient::setPause (const bool *on*)

Set pause.

Parameters

on	true for pause on; false for pause off.

6.107.3.41 void Coral::TraceClient::setPosition (const card64 *position*) [inline]

Set media position.

Parameters

position	New media position in nanoseconds.

6.107.3.42 void Coral::TraceClient::setSessionPriority (const int8 *priority*) [inline]

Set session priority.

Parameters

priority	Session priority.

6.107.3.43 void Coral::TraceClient::setStreamPriority (const int8 *priority*) [inline]

Set stream priority.

Parameters

priority	Stream priority.

6.107.3.44 void Coral::TraceClient::setWantedUtilization (const double *utilization*) [inline]

Set wanted utilization.

Parameters

utilization	Wanted utilization.

6.107.3.45 void Coral::TraceClient::stop ( )

Stop playing.

6.107.4 Member Data Documentation

6.107.4.1 card64 Coral::TraceClient::ChangeTimeStamp [private]

6.107.4.2 TraceDecoderRepository Coral::TraceClient::Decoders [private]

6.107.4.3 multimap<const cardinal,TraceDecoderInterface*> Coral::TraceClient::DecoderSet [private]

6.107.4.4 InternetFlow Coral::TraceClient::Flow [private]

6.107.4.5 bool Coral::TraceClient::IsPlaying [private]

6.107.4.6 card64 Coral::TraceClient::OldPosition [private]

6.107.4.7 InternetAddress Coral::TraceClient::OurAddress [private]

6.107.4.8 card32 Coral::TraceClient::OurSSRC [private]

6.107.4.9 RTPReceiver* Coral::TraceClient::Receiver [private]

6.107.4.10 InternetAddress Coral::TraceClient::ReceiverAddress [private]

6.107.4.11 Socket Coral::TraceClient::ReceiverSocket [private]

6.107.4.12 const card64 Coral::TraceClient::RestartPositionUpdateDelay = 5000000 [static, private]

6.107.4.13 RTCPSender* Coral::TraceClient::Sender [private]

6.107.4.14 Socket Coral::TraceClient::SenderSocket [private]

6.107.4.15 InternetAddress Coral::TraceClient::ServerAddress [private]

6.107.4.16 TraceClientAppPacket Coral::TraceClient::Status [private]

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

- traceclient.h
- traceclient.cc

6.108 Coral::TraceClientAppPacket Class Reference

Trace Client RTCP-SDES-APP-PRIV Packet.

`#include <traceclientapppacket.h>`

Public Types

- enum TraceClientAppMode { TCAS_UnknownCommand = 0, TCAS_Play = 1, TCAS_Pause = 2 }

Public Member Functions

- TraceClientAppPacket ()
- void translate ()
- void reset ()

Public Attributes

- card32 FormatID
- card16 SequenceNumber
- card16 PosChgSeqNumber
- card64 StartPosition

- card64 RestartPosition
- card64 WantedUtilization
- card32 Flags
- card64 MinWantedBandwidth
- card64 MaxWantedBandwidth
- card32 MaxTransferDelay
- card16 Status
- card16 Encoding
- int8 StreamPriority
- int8 SessionPriority
- card32 SessionDescriptor
- char MediaName [128]

Static Public Attributes

- static const card32 TraceClientFormatID = 0x64643554
- static const cardinal RTPTraceDefaultPort = 7100

6.108.1 Detailed Description

Trace Client RTCP-SDES-APP-PRIV Packet.

This class defines the packet format for the trace client's RTCP APP-PRIV messages.

Author

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

See also

TraceClient
TraceServer

6.108.2 Member Enumeration Documentation

6.108.2.1 enum Coral::TraceClientAppPacket::TraceClientAppMode

Definition of TraceClient commands in APP message.

Enumerator:

TCAS_UnknownCommand

TCAS_Play

TCAS_Pause

6.108.3 Constructor & Destructor Documentation

6.108.3.1 Coral::TraceClientAppPacket::TraceClientAppPacket ( )

Constructor.

6.108.4 Member Function Documentation

6.108.4.1 void Coral::TraceClientAppPacket::reset ( )

Reset report.

6.108.4.2 void Coral::TraceClientAppPacket::translate ( )

Translate byte order.

6.108.5 Member Data Documentation

6.108.5.1 card16 Coral::TraceClientAppPacket::Encoding

Encoding.

6.108.5.2 card32 Coral::TraceClientAppPacket::Flags

Flags.

6.108.5.3 card32 Coral::TraceClientAppPacket::FormatID

Packet ID.

6.108.5.4 card32 Coral::TraceClientAppPacket::MaxTransferDelay

Max wanted delay.

6.108.5.5 card64 Coral::TraceClientAppPacket::MaxWantedBandwidth

Maximum wanted bandwidth.

6.108.5.6 char Coral::TraceClientAppPacket::MediaName[128]

Media name, e.g. "TraceFiles/Test1.tdtf".

6.108.5.7 card64 Coral::TraceClientAppPacket::MinWantedBandwidth

Minimum wanted bandwidth.

6.108.5.8 card16 Coral::TraceClientAppPacket::PosChgSeqNumber

Sequence number for position changes.

6.108.5.9 card64 Coral::TraceClientAppPacket::RestartPosition

Position to start from if server has been restarted.

6.108.5.10 const cardinal Coral::TraceClientAppPacket::RTPTraceDefaultPort = 7100 [static]

RTP Trace Server default port.

6.108.5.11 card16 Coral::TraceClientAppPacket::SequenceNumber

Sequence number.

6.108.5.12 card32 Coral::TraceClientAppPacket::SessionDescriptor

Session descriptor.

6.108.5.13 int8 Coral::TraceClientAppPacket::SessionPriority

Session priority.

6.108.5.14 card64 Coral::TraceClientAppPacket::StartPosition

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

6.108.5.15 card16 Coral::TraceClientAppPacket::Status

Client status.

6.108.5.16 int8 Coral::TraceClientAppPacket::StreamPriority

Stream priority.

6.108.5.17 const card32 Coral::TraceClientAppPacket::TraceClientFormatID = 0x64643554 [static]

Packet ID for TraceClient RTCP APP message.

6.108.5.18 card64 Coral::TraceClientAppPacket::WantedUtilization

Wanted utilization;

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

- traceclientapppacket.h
- traceclientapppacket.cc

6.109 Coral::TraceConfiguration Struct Reference

Trace Configuration.

`#include <traceconfiguration.h>`

Public Member Functions

- void print (ostream &os, const bool utilizationOnly=false) const
- bool load (const char *fileName, const bool utilizationOnly=false)

Public Attributes

- double FrameRate
- char FramePattern [256]
- char InputName [128]
- card16 MediaType
- card16 MediaSubtype
- cardinal ExtLayers
- double FakeE1
- double FakeE2
- cardinal MaxBufferDelay
- card64 MinIntervalLength
- card64 MaxIntervalLength
- double RemappingCost
- card16 RUPoints
- card64 BandwidthThreshold
- double UtilizationThreshold
- card16 FrameRateUtilizationMaxConstants
- card16 FrameRateUtilizationConstants
- card16 FrameRateUtilizationType

- double FrameRateUtilizationWeight
- double FrameRateUtilizationConstant [MaxFrameRateUtilizationConstants]
- char Title [TDTFPrefix::MaxTitleLength]
- char Copyright [TDTFPrefix::MaxCopyrightLength]
- char Comment [TDTFPrefix::MaxCopyrightLength]
- char URL [TDTFPrefix::MaxURLLength]
- TraceLayerConfiguration Layer [MaxLayers]

Static Public Attributes

- static const card16 MaxFrameRateUtilizationConstants = 8
- static const cardinal MaxLayers = 16

6.109.1 Detailed Description

Trace Configuration.

This is a trace configuration.

Author

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

6.109.2 Member Function Documentation

6.109.2.1 bool Coral::TraceConfiguration::load (const char * *fileName*, const bool *utilizationOnly* = false)

Load configuration from file.

Parameters

fileName	File name of file to load.
utilitazion-Only	true to load utilization values only; false otherwise.

Returns

true, if load has been successful; false otherwise.

6.109.2.2 void Coral::TraceConfiguration::print (ostream & *os,* const bool *utilizationOnly* = `false`) const

Print configuration to given output stream.

Parameters

os	Output stream.
utilitazion-Only	true to print utilization values only; false otherwise.

6.109.3 Member Data Documentation

6.109.3.1 card64 Coral::TraceConfiguration::BandwidthThreshold

Bandwidth threshold.

6.109.3.2 char Coral::TraceConfiguration::Comment[TDTFPrefix::MaxCopyright-Length]

Comment.

6.109.3.3 char Coral::TraceConfiguration::Copyright[TDTFPrefix::MaxCopyright-Length]

Copyright.

6.109.3.4 cardinal Coral::TraceConfiguration::ExtLayers

Number of extension layers.

6.109.3.5 double Coral::TraceConfiguration::FakeE1

Fake factor for 1st extension layer: FrameSizeE1 = FakeE1 $*$ FrameSizeBase.

6.109.3.6 double Coral::TraceConfiguration::FakeE2

Fake factor for 2nd extension layer: FrameSizeE2 = FakeE2 $*$ FrameSizeBase.

6.109.3.7 char Coral::TraceConfiguration::FramePattern[256]

Frame pattern, e.g. "IBBPBBPBBPBBP".

6.109.3.8 double Coral::TraceConfiguration::FrameRate

Frame rate.

6.109.3.9 double Coral::TraceConfiguration::FrameRateUtilizationConstant[MaxFrameRateUtilizationConstants]

Array of frame rate utilization constants.

6.109.3.10 card16 Coral::TraceConfiguration::FrameRateUtilizationConstants

Number of frame rate utilization constants.

6.109.3.11 card16 Coral::TraceConfiguration::FrameRateUtilizationMaxConstants

Number of frame rate utilization constants.

6.109.3.12 card16 Coral::TraceConfiguration::FrameRateUtilizationType

Frame rate utility function type.

6.109.3.13 double Coral::TraceConfiguration::FrameRateUtilizationWeight

Frame rate utilization weight.

6.109.3.14 char Coral::TraceConfiguration::InputName[128]

Name of input trace file.

6.109.3.15 TraceLayerConfiguration Coral::TraceConfiguration::Layer[MaxLayers]

Array of layer configurations.

6.109.3.16 cardinal Coral::TraceConfiguration::MaxBufferDelay

Maximum buffer delay in microseconds.

6.109.3.17 const card16 Coral::TraceConfiguration::MaxFrameRateUtilization-Constants = 8 [static]

Maximum number of frame rate utilization constants.

6.109.3.18 card64 Coral::TraceConfiguration::MaxIntervalLength

Maximum interval length in microseconds.

6.109.3.19 const cardinal Coral::TraceConfiguration::MaxLayers = 16 [static]

Maximum number of layers.

6.109.3.20 card16 Coral::TraceConfiguration::MediaSubtype

Media subtype.

6.109.3.21 card16 Coral::TraceConfiguration::MediaType

Media type.

See also

TDTFPrefix::MediaTypes

6.109.3.22 card64 Coral::TraceConfiguration::MinIntervalLength

Minimum interval length in microseconds.

6.109.3.23 double Coral::TraceConfiguration::RemappingCost

Remapping cost.

6.109.3.24 card16 Coral::TraceConfiguration::RUPoints

Number of resource/utilization points.

6.109.3.25 char Coral::TraceConfiguration::Title[TDTFPrefix::MaxTitleLength]

Title.

6.109.3.26 char Coral::TraceConfiguration::URL[TDTFPrefix::MaxURLLength]

URL.

6.109.3.27 double Coral::TraceConfiguration::UtilizationThreshold

Utilization threshold.

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

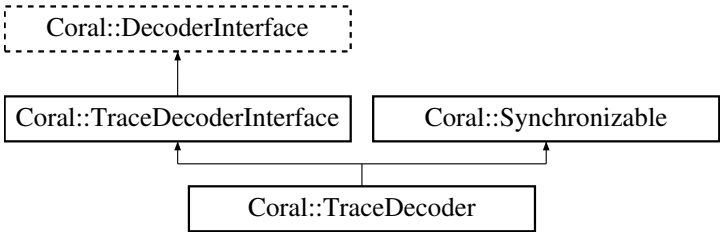
- traceconfiguration.h
- traceconfiguration.cc

6.110 Coral::TraceDecoder Class Reference

Trace Decoder.

`#include <tracedecoder.h>`

Inheritance diagram for Coral::TraceDecoder:



Public Member Functions

- TraceDecoder ()
- ~TraceDecoder ()
- 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)
- double getFrameRate () const

- double getUtilization () const
- Range< card64 > getBandwidth () const
- int8 getStreamPriority () const
- int8 getSessionPriority () const
- card32 getFlags () const

Private Attributes

- SeqNumValidator SeqNumber [RTPConstants::RTPMaxQualityLayers]
- cardinal FrameID [RTPConstants::RTPMaxQualityLayers]
- cardinal FrameSize [RTPConstants::RTPMaxQualityLayers]
- card64 Position
- card64 MaxPosition
- Range< card64 > Bandwidth
- cardinal MinBandwidth
- cardinal MaxBandwidth
- double FrameRate
- double Utilization
- card32 Flags
- MediaInfo Media
- card8 ErrorCode
- int8 StreamPriority
- int8 SessionPriority

6.110.1 Detailed Description

Trace Decoder.

This class is a trace decoder.

Author

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

6.110.2 Constructor & Destructor Documentation

6.110.2.1 Coral::TraceDecoder::TraceDecoder ( )

Constructor.

6.110.2.2 Coral::TraceDecoder::~TraceDecoder ( )

Destructor.

6.110.3 Member Function Documentation

6.110.3.1 void Coral::TraceDecoder::activate ( ) [virtual]

activate() implementation of DecoderInterface.

See also

DecoderInterface::activate

Implements Coral::DecoderInterface.

6.110.3.2 bool Coral::TraceDecoder::checkNextPacket (DecoderPacket * *decoderPacket*) [virtual]

checkNextPacket() implementation of DecoderInterface.

See also

DecoderInterface::checkNextPacket

Implements Coral::DecoderInterface.

6.110.3.3 void Coral::TraceDecoder::deactivate ( ) [virtual]

deactivate() implementation of DecoderInterface.

See also

DecoderInterface::deactivate

Implements Coral::DecoderInterface.

6.110.3.4 Range< card64 > Coral::TraceDecoder::getBandwidth ( ) const [virtual]

getBandwidth() implementation of TraceEncoderInterface.

See also

TraceEncoderInterface::getBandwidth

Implements Coral::TraceDecoderInterface.

6.110.3.5 card8 Coral::TraceDecoder::getErrorCode ( ) const [virtual]

getErrorCode() implementation of DecoderInterface.

See also

DecoderInterface::getErrorCode

Implements Coral::DecoderInterface.

6.110.3.6 card32 Coral::TraceDecoder::getFlags ( ) const [virtual]

getFlags() implementation of TraceEncoderInterface.

See also

TraceEncoderInterface::getFlags

Implements Coral::TraceDecoderInterface.

6.110.3.7 double Coral::TraceDecoder::getFrameRate ( ) const [virtual]

getFrameRate() implementation of TraceEncoderInterface.

See also

TraceEncoderInterface::getFrameRate

Implements Coral::TraceDecoderInterface.

6.110.3.8 card64 Coral::TraceDecoder::getMaxPosition ( ) const [virtual]

getMaxPosition() implementation of DecoderInterface.

See also

DecoderInterface::getMaxPosition

Implements Coral::DecoderInterface.

6.110.3.9 void Coral::TraceDecoder::getMediaInfo (MediaInfo & *mediaInfo*) const [virtual]

getMediaInfo() implementation of DecoderInterface.

See also

DecoderInterface::getMediaInfo

Implements Coral::DecoderInterface.

6.110.3.10 card64 Coral::TraceDecoder::getPosition ( ) const [virtual]

getPosition() implementation of DecoderInterface.

See also

DecoderInterface::getPosition

Implements Coral::DecoderInterface.

6.110.3.11 int8 Coral::TraceDecoder::getSessionPriority ( ) const [virtual]

getSessionPriority() implementation of TraceEncoderInterface.

See also

TraceEncoderInterface::getSessionPriority

Implements Coral::TraceDecoderInterface.

6.110.3.12 int8 Coral::TraceDecoder::getStreamPriority ( ) const [virtual]

getStreamPriority() implementation of TraceEncoderInterface.

See also

TraceEncoderInterface::getStreamPriority

Implements Coral::TraceDecoderInterface.

6.110.3.13 const card16 Coral::TraceDecoder::getTypeID ( ) const [virtual]

getTypeID() implementation of DecoderInterface.

See also

DecoderInterface::getTypeID

Implements Coral::DecoderInterface.

6.110.3.14 const char * Coral::TraceDecoder::getTypeName ( ) const [virtual]

getTypeName implementation of DecoderInterface.

See also

DecoderInterface::getTypeName

Implements Coral::DecoderInterface.

6.110.3.15 double Coral::TraceDecoder::getUtilization ( ) const [virtual]

getUtilization() implementation of TraceEncoderInterface.

See also

TraceEncoderInterface::getUtilization

Implements Coral::TraceDecoderInterface.

6.110.3.16 void Coral::TraceDecoder::handleNextPacket (const DecoderPacket * *decoderPacket*) [virtual]

handleNextPacket() implementation of DecoderInterface.

See also

DecoderInterface::handleNextPacket

Implements Coral::DecoderInterface.

6.110.3.17 void Coral::TraceDecoder::reset ( ) [virtual]

reset() implementation of DecoderInterface.

See also

DecoderInterface::reset

Implements Coral::DecoderInterface.

6.110.4 Member Data Documentation

6.110.4.1 Range<card64> Coral::TraceDecoder::Bandwidth [private]

6.110.4.2 card8 Coral::TraceDecoder::ErrorCode [private]

6.110.4.3 card32 Coral::TraceDecoder::Flags [private]

6.110.4.4 cardinal Coral::TraceDecoder::FrameID[RTPConstants::RTPMaxQualityLayers] [private]

6.110.4.5 double Coral::TraceDecoder::FrameRate [private]

6.110.4.6 cardinal Coral::TraceDecoder::FrameSize[RTPConstants::RTPMaxQualityLayers] [private]

6.110.4.7 cardinal Coral::TraceDecoder::MaxBandwidth [private]

6.110.4.8 card64 Coral::TraceDecoder::MaxPosition [private]

6.110.4.9 MediaInfo Coral::TraceDecoder::Media [private]

6.110.4.10 cardinal Coral::TraceDecoder::MinBandwidth [private]

6.110.4.11 card64 Coral::TraceDecoder::Position [private]

6.110.4.12 SeqNumValidator Coral::TraceDecoder::SeqNumber[RTPConstants::RTPMaxQualityLayers] [private]

6.110.4.13 int8 Coral::TraceDecoder::SessionPriority [private]

6.110.4.14 int8 Coral::TraceDecoder::StreamPriority [private]

6.110.4.15 double Coral::TraceDecoder::Utilization [private]

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

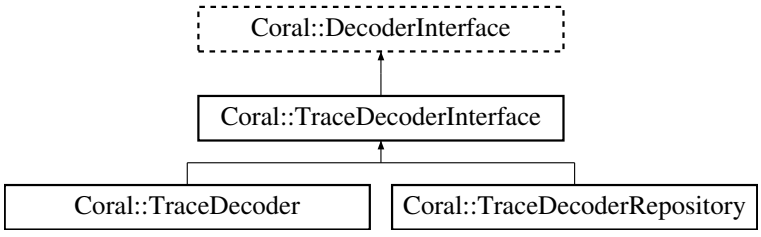
- tracedecoder.h
- tracedecoder.cc

6.111 Coral::TraceDecoderInterface Class Reference

Trace Decoder Interface.

`#include <tracedecoderinterface.h>`

Inheritance diagram for Coral::TraceDecoderInterface:



Public Member Functions

- virtual double getFrameRate () const =0
- virtual double getUtilization () const =0
- virtual cardinal getFlags () const =0

- virtual Range< card64 > getBandwidth () const =0
- virtual int8 getStreamPriority () const =0
- virtual int8 getSessionPriority () const =0

6.111.1 Detailed Description

Trace Decoder Interface.

This class is the interface for a trace decoder.

Author

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

6.111.2 Member Function Documentation

6.111.2.1 virtual Range<card64> Coral::TraceDecoderInterface::getBandwidth ( ) const [pure virtual]

Get bandwidth range.

Returns

Bandwidth range.

Implemented in Coral::TraceDecoderRepository, and Coral::TraceDecoder.

6.111.2.2 virtual cardinal Coral::TraceDecoderInterface::getFlags ( ) const [pure virtual]

Get flags.

Returns

flags.

Implemented in Coral::TraceDecoderRepository, and Coral::TraceDecoder.

6.111.2.3 virtual double Coral::TraceDecoderInterface::getFrameRate ( ) const [pure virtual]

Set frameRate.

Parameters

frameRate	FrameRate.

Returns

FrameRate set.

Implemented in Coral::TraceDecoderRepository, and Coral::TraceDecoder.

6.111.2.4 virtual int8 Coral::TraceDecoderInterface::getSessionPriority ( ) const [pure virtual]

Get session priority.

Returns

Session priority.

Implemented in Coral::TraceDecoderRepository, and Coral::TraceDecoder.

6.111.2.5 virtual int8 Coral::TraceDecoderInterface::getStreamPriority ( ) const [pure virtual]

Get stream priority.

Returns

Stream priority.

Implemented in Coral::TraceDecoderRepository, and Coral::TraceDecoder.

6.111.2.6 virtual double Coral::TraceDecoderInterface::getUtilization ( ) const [pure virtual]

Get utilization.

Returns

Utilization (out of [0,1]).

Implemented in Coral::TraceDecoderRepository, and Coral::TraceDecoder.

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

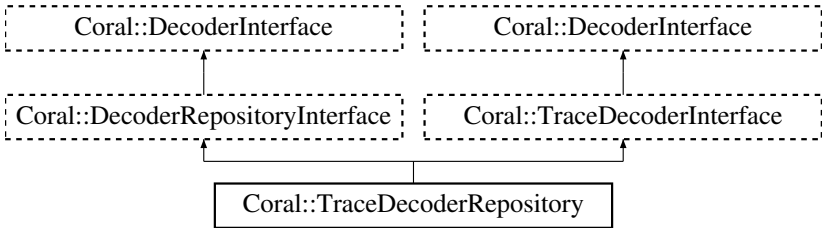
- tracedecoderinterface.h

6.112 Coral::TraceDecoderRepository Class Reference

Trace Decoder Repository.

`#include <tracedecoderrepository.h>`

Inheritance diagram for Coral::TraceDecoderRepository:



Public Member Functions

- TraceDecoderRepository ()
- ~TraceDecoderRepository ()
- bool addDecoder (TraceDecoderInterface *decoder)
- void removeDecoder (TraceDecoderInterface *decoder)
- bool selectDecoderForTypeID (const card16 typeID)
- void setAutoDelete (const bool on)
- DecoderInterface * getCurrentDecoder () const
- TraceDecoderInterface * getCurrentTraceDecoder () 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)
- double getFrameRate () const
- double getUtilization () const
- card32 getFlags () const
- Range< card64 > getBandwidth () const
- int8 getStreamPriority () const
- int8 getSessionPriority () const

Private Attributes

- multimap< const card16, TraceDecoderInterface * > TraceDecoderRepository
- TraceDecoderInterface * Decoder
- bool AutoDelete

6.112.1 Detailed Description

Trace Decoder Repository.

This class is a repository for trace decoders.

Author

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

6.112.2 Constructor & Destructor Documentation

6.112.2.1 Coral::TraceDecoderRepository::TraceDecoderRepository ( )

Constructor.

6.112.2.2 Coral::TraceDecoderRepository::~TraceDecoderRepository ( )

Destructor.

6.112.3 Member Function Documentation

6.112.3.1 void Coral::TraceDecoderRepository::activate ( ) [virtual]

activate() implementation of DecoderInterface.

See also

DecoderInterface::activate

Implements Coral::DecoderInterface.

6.112.3.2 bool Coral::TraceDecoderRepository::addDecoder (TraceDecoderInterface * *decoder*)

Add trace decoder to repository.

Parameters

decoder	New trace decoder to be added.

Returns

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

6.112.3.3 bool Coral::TraceDecoderRepository::checkNextPacket (DecoderPacket * *decoderPacket*) [virtual]

checkNextPacket() implementation of DecoderInterface.

See also

DecoderInterface::checkNextPacket

Implements Coral::DecoderInterface.

6.112.3.4 void Coral::TraceDecoderRepository::deactivate ( ) [virtual]

deactivate() implementation of DecoderInterface.

See also

DecoderInterface::deactivate

Implements Coral::DecoderInterface.

6.112.3.5 Range< card64 > Coral::TraceDecoderRepository::getBandwidth ( ) const [virtual]

getBandwidth() implementation of TraceEncoderInterface.

See also

TraceEncoderInterface::getBandwidth

Implements Coral::TraceDecoderInterface.

6.112.3.6 DecoderInterface * Coral::TraceDecoderRepository::getCurrentDecoder ( ) const [virtual]

getCurrentDecoder() implementation of DecoderRepositoryInterface.

See also

DecoderRepositoryInterface::getCurrentDecoder

Implements Coral::DecoderRepositoryInterface.

6.112.3.7 TraceDecoderInterface * Coral::TraceDecoderRepository::getCurrentTraceDecoder ( ) const

Get TraceDecoderInterface of the current decoder.

Returns

Current decoder's TraceDecoderInterface.

6.112.3.8 card8 Coral::TraceDecoderRepository::getErrorCode ( ) const [virtual]

getErrorCode() implementation of DecoderInterface.

See also

DecoderInterface::getErrorCode

Implements Coral::DecoderInterface.

6.112.3.9 card32 Coral::TraceDecoderRepository::getFlags ( ) const [virtual]

getFlags() implementation of TraceEncoderInterface.

See also

TraceEncoderInterface::getFlags

Implements Coral::TraceDecoderInterface.

6.112.3.10 double Coral::TraceDecoderRepository::getFrameRate ( ) const [virtual]

getFrameRate() implementation of TraceEncoderInterface.

See also

TraceEncoderInterface::getFrameRate

Implements Coral::TraceDecoderInterface.

6.112.3.11 card64 Coral::TraceDecoderRepository::getMaxPosition ( ) const [virtual]

getMaxPosition() implementation of DecoderInterface.

See also

DecoderInterface::getMaxPosition

Implements Coral::DecoderInterface.

6.112.3.12 void Coral::TraceDecoderRepository::getMediaInfo (MediaInfo & *mediaInfo*) const [virtual]

getMediaInfo() implementation of DecoderInterface.

See also

DecoderInterface::getMediaInfo

Implements Coral::DecoderInterface.

6.112.3.13 card64 Coral::TraceDecoderRepository::getPosition ( ) const [virtual]

getPosition() implementation of DecoderInterface.

See also

DecoderInterface::getPosition

Implements Coral::DecoderInterface.

6.112.3.14 int8 Coral::TraceDecoderRepository::getSessionPriority ( ) const [virtual]

getSessionPriority() implementation of TraceEncoderInterface.

See also

TraceEncoderInterface::getSessionPriority

Implements Coral::TraceDecoderInterface.

6.112.3.15 int8 Coral::TraceDecoderRepository::getStreamPriority ( ) const [virtual]

getStreamPriority() implementation of TraceEncoderInterface.

See also

TraceEncoderInterface::getStreamPriority

Implements Coral::TraceDecoderInterface.

6.112.3.16 const card16 Coral::TraceDecoderRepository::getTypeID ( ) const [virtual]

getTypeID() implementation of DecoderInterface.

See also

DecoderInterface::getTypeID

Implements Coral::DecoderInterface.

6.112.3.17 const char * Coral::TraceDecoderRepository::getTypeName ( ) const [virtual]

getTypeName implementation of DecoderInterface.

See also

DecoderInterface::getTypeName

Implements Coral::DecoderInterface.

6.112.3.18 double Coral::TraceDecoderRepository::getUtilization ( ) const [virtual]

getUtilization() implementation of TraceEncoderInterface.

See also

TraceEncoderInterface::getUtilization

Implements Coral::TraceDecoderInterface.

6.112.3.19 void Coral::TraceDecoderRepository::handleNextPacket (const DecoderPacket * *decoderPacket*) [virtual]

handleNextPacket() implementation of DecoderInterface.

See also

DecoderInterface::handleNextPacket

Implements Coral::DecoderInterface.

6.112.3.20 void Coral::TraceDecoderRepository::removeDecoder (TraceDecoderInterface * *decoder*)

Remove trace decoder from repository.

Parameters

decoder	Trace decoder to be removed.

6.112.3.21 void Coral::TraceDecoderRepository::reset ( ) [virtual]

reset() implementation of DecoderInterface.

See also

DecoderInterface::reset

Implements Coral::DecoderInterface.

6.112.3.22 bool Coral::TraceDecoderRepository::selectDecoderForTypeID (const card16 *typeID*) [virtual]

selectDecoderForTypeID() implementation of DecoderRepositoryInterface.

See also

DecoderRepositoryInterface::selectDecoderForTypeID

Implements Coral::DecoderRepositoryInterface.

6.112.3.23 void Coral::TraceDecoderRepository::setAutoDelete (const bool *on*) [inline]

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

6.112.4 Member Data Documentation

6.112.4.1 bool Coral::TraceDecoderRepository::AutoDelete [private]

6.112.4.2 TraceDecoderInterface* Coral::TraceDecoderRepository::Decoder [private]

6.112.4.3 Coral::TraceDecoderRepository::TraceDecoderRepository [private]

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

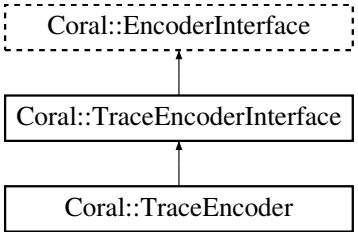
- tracedecoderrepository.h
- tracedecoderrepository.cc

6.113 Coral::TraceEncoder Class Reference

Trace Encoder.

`#include <traceencoder.h>`

Inheritance diagram for Coral::TraceEncoder:



Public Member Functions

- TraceEncoder (TDTFMediaReader ∗audioReader)
- ∼TraceEncoder ()
- 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)
- AbstractQoSDescription ∗ getQoSDescription (const cardinal pktHeaderSize, const cardinal pktMaxSize, const card64 offset)
- void updateQuality (const AbstractQoSDescription ∗aqd)
- double getFrameRate () const
- void setFrameRate (const double frameRate)
- double getWantedUtilization () const
- void setWantedUtilization (const double utilization)
- int8 getStreamPriority () const
- void setStreamPriority (const int8 priority)
- int8 getSessionPriority () const
- void setSessionPriority (const int8 priority)
- double getMaxTransferDelay () const
- void setMaxTransferDelay (const double delay)
- card32 getFlags () const
- void setFlags (const card32 flags)
- card64 getMinWantedBandwidth () const
- card64 getMaxWantedBandwidth () const
- void setMinWantedBandwidth (const card64 bandwidth)
- void setMaxWantedBandwidth (const card64 bandwidth)

Public Attributes

- double CurrentUtilization
- card64 CurrentBandwidth

Private Attributes

- TDTFMediaReader * Source
- double ManagerFrameRate
- double ManagerScaleFactor [RTPConstants::RTPMaxQualityLayers]
- cardinal ManagerFrameSizeLimit [RTPConstants::RTPMaxQualityLayers]
- double MaxTransferDelay
- card64 MinWantedBandwidth
- card64 MaxWantedBandwidth
- double WantedUtilization
- double FrameFrameRate
- double FrameUtilization
- card64 FramePosition
- card64 FrameMaxPosition
- card64 FrameBandwidth
- card32 FrameFlags
- card32 Flags
- cardinal FrameSize [RTPConstants::RTPMaxQualityLayers]
- cardinal Offset [RTPConstants::RTPMaxQualityLayers]
- cardinal FrameID [RTPConstants::RTPMaxQualityLayers]
- cardinal Layers
- cardinal SendError
- card8 ErrorCode
- int8 StreamPriority
- int8 SessionPriority
- bool Scaled
- bool Paused

6.113.1 Detailed Description

Trace Encoder.

This class is a trace encoder.

Author

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

6.113.2 Constructor & Destructor Documentation

6.113.2.1 Coral::TraceEncoder::TraceEncoder (TDTFMediaReader * *audioReader*)

Constructor for the trace encoder.

Parameters

traceReader	TDTFReader for the trace input.

6.113.2.2 Coral::TraceEncoder::~TraceEncoder ( )

Destructor.

6.113.3 Member Function Documentation

6.113.3.1 void Coral::TraceEncoder::activate ( ) [virtual]

activate() implementation of EncoderInterface.

See also

EncoderInterface::activate

Implements Coral::EncoderInterface.

6.113.3.2 bool Coral::TraceEncoder::checkInterval (card64 & *time,* bool & *newRUList*) [virtual]

checkInterval() implementation of EncoderInterface.

See also

EncoderInterface::checkInterval

Implements Coral::EncoderInterface.

6.113.3.3 void Coral::TraceEncoder::deactivate ( ) [virtual]

deactivate() implementation of EncoderInterface.

See also

EncoderInterface::deactivate

Implements Coral::EncoderInterface.

6.113.3.4 card32 Coral::TraceEncoder::getFlags ( ) const [virtual]

getFlags() implementation of TraceEncoderInterface.

See also

TraceEncoderInterface::getFlags

Implements Coral::TraceEncoderInterface.

6.113.3.5 double Coral::TraceEncoder::getFrameRate ( ) const

getFrameRate() implementation of TraceEncoderInterface.

See also

TraceEncoderInterface::getFrameRate

6.113.3.6 double Coral::TraceEncoder::getMaxTransferDelay ( ) const [virtual]

getMaxTransferDelay() implementation of TraceEncoderInterface.

See also

TraceEncoderInterface::getMaxTransferDelay

Implements Coral::TraceEncoderInterface.

6.113.3.7 card64 Coral::TraceEncoder::getMaxWantedBandwidth ( ) const [virtual]

getMaxWantedBandwidth() implementation of TraceEncoderInterface.

See also

TraceEncoderInterface::getMaxWantedBandwidth

Implements Coral::TraceEncoderInterface.

6.113.3.8 card64 Coral::TraceEncoder::getMinWantedBandwidth ( ) const [virtual]

getMinWantedBandwidth() implementation of TraceEncoderInterface.

See also

TraceEncoderInterface::getMinWantedBandwidth

Implements Coral::TraceEncoderInterface.

6.113.3.9 cardinal Coral::TraceEncoder::getNextPacket (EncoderPacket * *encoderPacket*) [virtual]

getNextPacket() implementation of EncoderInterface.

See also

EncoderInterface::getNextPacket

Implements Coral::EncoderInterface.

6.113.3.10 AbstractQoSDescription * Coral::TraceEncoder::getQoSDescription (const cardinal *pktHeaderSize,* const cardinal *pktMaxSize,* const card64 *offset*) [virtual]

getQoSDescription() implementation of EncoderInterface.

See also

EncoderInterface::getQoSDescription

Implements Coral::EncoderInterface.

6.113.3.11 int8 Coral::TraceEncoder::getSessionPriority ( ) const [inline, virtual]

getSessionPriority() implementation of TraceEncoderInterface.

See also

TraceEncoderInterface::getSessionPriority

Implements Coral::TraceEncoderInterface.

6.113.3.12 int8 Coral::TraceEncoder::getStreamPriority ( ) const [inline, virtual]

getStreamPriority() implementation of TraceEncoderInterface.

See also

TraceEncoderInterface::getStreamPriority

Implements Coral::TraceEncoderInterface.

6.113.3.13 const card16 Coral::TraceEncoder::getTypeID ( ) const [virtual]

getTypeID() implementation of EncoderInterface.

See also

EncoderInterface::getTypeID

Implements Coral::EncoderInterface.

6.113.3.14 const char * Coral::TraceEncoder::getTypeName ( ) const [virtual]

getTypeName implementation of EncoderInterface.

See also

EncoderInterface::getTypeName

Implements Coral::EncoderInterface.

6.113.3.15 double Coral::TraceEncoder::getWantedUtilization ( ) const [virtual]

getWantedUtilization() implementation of TraceEncoderInterface.

See also

TraceEncoderInterface::getWantedUtilization

Implements Coral::TraceEncoderInterface.

6.113.3.16 bool Coral::TraceEncoder::prepareNextFrame (const cardinal *headerSize*, const cardinal *maxPacketSize*, const cardinal *flags*) [virtual]

prepareNextFrame() implementation of EncoderInterface.

See also

EncoderInterface::prepareNextFrame

Implements Coral::EncoderInterface.

6.113.3.17 void Coral::TraceEncoder::reset ( ) [virtual]

reset() implementation of EncoderInterface.

See also

EncoderInterface::reset

Implements Coral::EncoderInterface.

6.113.3.18 void Coral::TraceEncoder::setFlags (const card32 *flags*) [virtual]

setFlags() implementation of TraceEncoderInterface.

See also

TraceEncoderInterface::setFlags

Implements Coral::TraceEncoderInterface.

6.113.3.19 void Coral::TraceEncoder::setFrameRate (const double *frameRate*)

setFrameRate() implementation of TraceEncoderInterface.

See also

TraceEncoderInterface::setFrameRate

6.113.3.20 void Coral::TraceEncoder::setMaxTransferDelay (const double *delay*) [virtual]

setMaxTransferDelay() implementation of TraceEncoderInterface.

See also

TraceEncoderInterface::setMaxWanted

Implements Coral::TraceEncoderInterface.

6.113.3.21 void Coral::TraceEncoder::setMaxWantedBandwidth (const card64 *bandwidth*) [virtual]

setMaxWantedBandwidth() implementation of TraceEncoderInterface.

See also

TraceEncoderInterface::setMaxWantedBandwidth

Implements Coral::TraceEncoderInterface.

6.113.3.22 void Coral::TraceEncoder::setMinWantedBandwidth (const card64 *bandwidth*) [virtual]

setMinWantedBandwidth() implementation of TraceEncoderInterface.

See also

TraceEncoderInterface::setMinWantedBandwidth

Implements Coral::TraceEncoderInterface.

6.113.3.23 void Coral::TraceEncoder::setSessionPriority (const int8 *priority*) [inline, virtual]

setSessionPriority() implementation of TraceEncoderInterface.

See also

TraceEncoderInterface::setSessionPriority

Implements Coral::TraceEncoderInterface.

6.113.3.24 void Coral::TraceEncoder::setStreamPriority (const int8 *priority*) [inline, virtual]

setStreamPriority() implementation of TraceEncoderInterface.

See also

TraceEncoderInterface::setStreamPriority

Implements Coral::TraceEncoderInterface.

6.113.3.25 void Coral::TraceEncoder::setWantedUtilization (const double *utilization*) [virtual]

setWantedUtilization() implementation of TraceEncoderInterface.

See also

TraceEncoderInterface::setWantedUtilization

Implements Coral::TraceEncoderInterface.

6.113.3.26 void Coral::TraceEncoder::updateQuality (const AbstractQoSDescription * *aqd*) [virtual]

updateQuality() implementation of EncoderInterface.

See also

EncoderInterface::updateQuality

Implements Coral::EncoderInterface.

6.113.4 Member Data Documentation

6.113.4.1 card64 Coral::TraceEncoder::CurrentBandwidth

6.113.4.2 double Coral::TraceEncoder::CurrentUtilization

6.113.4.3 card8 Coral::TraceEncoder::ErrorCode [private]

6.113.4.4 card32 Coral::TraceEncoder::Flags [private]

6.113.4.5 card64 Coral::TraceEncoder::FrameBandwidth [private]

6.113.4.6 card32 Coral::TraceEncoder::FrameFlags [private]

6.113.4.7 double Coral::TraceEncoder::FrameFrameRate [private]

6.113.4.8 cardinal Coral::TraceEncoder::FrameID[RTPConstants::RTPMaxQualityLayers] [private]

6.113.4.9 card64 Coral::TraceEncoder::FrameMaxPosition [private]

6.113.4.10 card64 Coral::TraceEncoder::FramePosition [private]

6.113.4.11 cardinal Coral::TraceEncoder::FrameSize[RTPConstants::RTPMaxQualityLayers] [private]

6.113.4.12 double Coral::TraceEncoder::FrameUtilization [private]

6.113.4.13 cardinal Coral::TraceEncoder::Layers [private]

6.113.4.14 double Coral::TraceEncoder::ManagerFrameRate [private]

6.113.4.15 cardinal Coral::TraceEncoder::ManagerFrameSizeLimit[RTPConstants::RTPMaxQualityLayers] [private]

6.113.4.16 double Coral::TraceEncoder::ManagerScaleFactor[RTPConstants::RTPMaxQualityLayers] [private]

6.113.4.17 double Coral::TraceEncoder::MaxTransferDelay [private]

6.113.4.18 card64 Coral::TraceEncoder::MaxWantedBandwidth [private]

6.113.4.19 card64 Coral::TraceEncoder::MinWantedBandwidth [private]

6.113.4.20 cardinal Coral::TraceEncoder::Offset[RTPConstants::RTPMaxQualityLayers] [private]

6.113.4.21 bool Coral::TraceEncoder::Paused [private]

6.113.4.22 bool Coral::TraceEncoder::Scaled [private]

6.113.4.23 cardinal Coral::TraceEncoder::SendError [private]

6.113.4.24 int8 Coral::TraceEncoder::SessionPriority [private]

6.113.4.25 TDTFMediaReader* Coral::TraceEncoder::Source [private]

6.113.4.26 int8 Coral::TraceEncoder::StreamPriority [private]

6.113.4.27 double Coral::TraceEncoder::WantedUtilization [private]

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

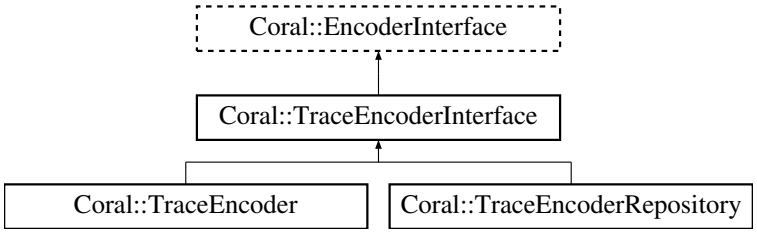
- traceencoder.h
- traceencoder.cc

6.114 Coral::TraceEncoderInterface Class Reference

Trace Encoder Interface.

`#include <traceencoderinterface.h>`

Inheritance diagram for Coral::TraceEncoderInterface:



Public Member Functions

- virtual double getWantedUtilization () const =0
- virtual void setWantedUtilization (const double utilization)=0
- virtual int8 getStreamPriority () const =0
- virtual void setStreamPriority (const int8 priority)=0
- virtual int8 getSessionPriority () const =0
- virtual void setSessionPriority (const int8 priority)=0
- virtual double getMaxTransferDelay () const =0
- virtual void setMaxTransferDelay (const double delay)=0
- virtual cardinal getFlags () const =0

- virtual void setFlags (const card32 flags)=0
- virtual card64 getMinWantedBandwidth () const =0
- virtual card64 getMaxWantedBandwidth () const =0
- virtual void setMinWantedBandwidth (const card64 bandwidth)=0
- virtual void setMaxWantedBandwidth (const card64 bandwidth)=0

6.114.1 Detailed Description

Trace Encoder Interface.

This class is the interface for a trace encoder.

Author

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

6.114.2 Member Function Documentation

6.114.2.1 virtual cardinal Coral::TraceEncoderInterface::getFlags ( ) const [pure virtual]

Get flags.

Returns

flags.

Implemented in Coral::TraceEncoderRepository, and Coral::TraceEncoder.

6.114.2.2 virtual double Coral::TraceEncoderInterface::getMaxTransferDelay ( ) const [pure virtual]

Get maximum transfer delay.

Returns

Maximum transfer delay.

Implemented in Coral::TraceEncoderRepository, and Coral::TraceEncoder.

6.114.2.3 virtual card64 Coral::TraceEncoderInterface::getMaxWantedBandwidth ( ) const [pure virtual]

Get maximum wanted bandwidth.

Returns

Maximum wanted bandwidth.

Implemented in Coral::TraceEncoderRepository, and Coral::TraceEncoder.

6.114.2.4 virtual card64 Coral::TraceEncoderInterface::getMinWantedBandwidth ( ) const [pure virtual]

Get minimum wanted bandwidth.

Returns

Minimum wanted bandwidth.

Implemented in Coral::TraceEncoderRepository, and Coral::TraceEncoder.

6.114.2.5 virtual int8 Coral::TraceEncoderInterface::getSessionPriority ( ) const [pure virtual]

Get session priority.

Returns

Session priority.

Implemented in Coral::TraceEncoderRepository, and Coral::TraceEncoder.

6.114.2.6 virtual int8 Coral::TraceEncoderInterface::getStreamPriority ( ) const [pure virtual]

Get stream priority.

Returns

Stream priority.

Implemented in Coral::TraceEncoderRepository, and Coral::TraceEncoder.

6.114.2.7 virtual double Coral::TraceEncoderInterface::getWantedUtilization ( ) const [pure virtual]

Get wanted utilization.

Returns

Wanted utilization (out of [0,1]).

Implemented in Coral::TraceEncoderRepository, and Coral::TraceEncoder.

6.114.2.8 virtual void Coral::TraceEncoderInterface::setFlags (const card32 *flags*) [pure virtual]

Set flags.

Parameters

flags	Flags.

Implemented in Coral::TraceEncoderRepository, and Coral::TraceEncoder.

6.114.2.9 virtual void Coral::TraceEncoderInterface::setMaxTransferDelay (const double *delay*) [pure virtual]

Set maximum tranfer delay.

Parameters

delay	Maximum transfer delay.

Implemented in Coral::TraceEncoderRepository, and Coral::TraceEncoder.

6.114.2.10 virtual void Coral::TraceEncoderInterface::setMaxWantedBandwidth (const card64 *bandwidth*) [pure virtual]

Set maximum wanted bandwidth.

Parameters

wanted	bandwidth Maximum wanted bandwidth.

Implemented in Coral::TraceEncoderRepository, and Coral::TraceEncoder.

6.114.2.11 virtual void Coral::TraceEncoderInterface::setMinWantedBandwidth (const card64 *bandwidth*) [pure virtual]

Set minimum wanted bandwidth.

Parameters

wanted	bandwidth Minimum wanted bandwidth.

Implemented in Coral::TraceEncoderRepository, and Coral::TraceEncoder.

6.114.2.12 virtual void Coral::TraceEncoderInterface::setSessionPriority (const int8 *priority*) [pure virtual]

Set session priority.

Parameters

priority	Session priority.

Implemented in Coral::TraceEncoderRepository, and Coral::TraceEncoder.

6.114.2.13 virtual void Coral::TraceEncoderInterface::setStreamPriority (const int8 *priority*) [pure virtual]

Set stream priority.

Parameters

priority	Stream priority.

Implemented in Coral::TraceEncoderRepository, and Coral::TraceEncoder.

6.114.2.14 virtual void Coral::TraceEncoderInterface::setWantedUtilization (const double *utilization*) [pure virtual]

Set wanted utilization.

Parameters

utilization	Wanted utilization (out of [0,1]).

Implemented in Coral::TraceEncoderRepository, and Coral::TraceEncoder.

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

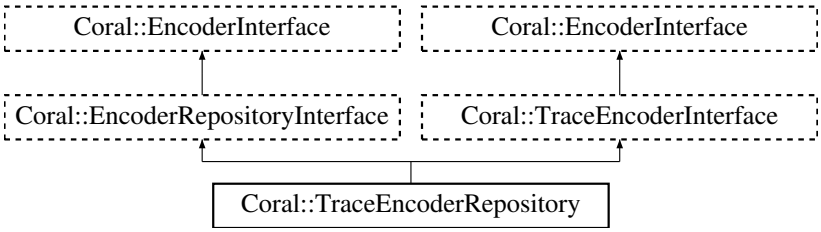
- traceencoderinterface.h

6.115 Coral::TraceEncoderRepository Class Reference

Trace Encoder Repository.

`#include <traceencoderrepository.h>`

Inheritance diagram for Coral::TraceEncoderRepository:



Public Member Functions

- TraceEncoderRepository ()
- ~TraceEncoderRepository ()
- bool addEncoder (TraceEncoderInterface *encoder)
- void removeEncoder (TraceEncoderInterface *encoder)
- bool selectEncoderForTypeID (const card16 typeID)
- void setAutoDelete (const bool on)
- EncoderInterface * getCurrentEncoder () const
- TraceEncoderInterface * getCurrentTraceEncoder () 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 maxPacket-Size, const cardinal flags)
- cardinal getNextPacket (EncoderPacket *encoderPacket)
- AbstractQoSDescription * getQoSDescription (const cardinal pktHeaderSize, const cardinal pktMaxSize, const card64 position)
- void updateQuality (const AbstractQoSDescription *aqd)
- double getWantedUtilization () const
- void setWantedUtilization (const double utilization)
- int8 getStreamPriority () const
- void setStreamPriority (const int8 priority)
- int8 getSessionPriority () const
- void setSessionPriority (const int8 priority)
- double getMaxTransferDelay () const
- void setMaxTransferDelay (const double delay)
- card32 getFlags () const
- void setFlags (const card32 flags)
- card64 getMinWantedBandwidth () const
- card64 getMaxWantedBandwidth () const
- void setMinWantedBandwidth (const card64 bandwidth)
- void setMaxWantedBandwidth (const card64 bandwidth)

Private Attributes

- multimap< const card16, TraceEncoderInterface * > TraceEncoderRepository
- TraceEncoderInterface * Encoder
- bool AutoDelete

6.115.1 Detailed Description

Trace Encoder Repository.

This class is a repository for trace encoders.

Author

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

6.115.2 Constructor & Destructor Documentation

6.115.2.1 Coral::TraceEncoderRepository::TraceEncoderRepository ( )

Constructor.

6.115.2.2 Coral::TraceEncoderRepository::~TraceEncoderRepository ( )

Destructor.

6.115.3 Member Function Documentation

6.115.3.1 void Coral::TraceEncoderRepository::activate ( ) [virtual]

activate() implementation of EncoderInterface.

See also

EncoderInterface::activate

Implements Coral::EncoderInterface.

6.115.3.2 bool Coral::TraceEncoderRepository::addEncoder (TraceEncoderInterface * *encoder*)

Add trace encoder to repository.

Parameters

encoder	New trace encoder to be added.

Returns

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

6.115.3.3 bool Coral::TraceEncoderRepository::checkInterval (card64 & *time,* bool & *newRUList*) [virtual]

checkInterval() implementation of EncoderInterface.

See also

EncoderInterface::checkInterval

Implements Coral::EncoderInterface.

6.115.3.4 void Coral::TraceEncoderRepository::deactivate ( ) [virtual]

deactivate() implementation of EncoderInterface.

See also

EncoderInterface::deactivate

Implements Coral::EncoderInterface.

6.115.3.5 EncoderInterface * Coral::TraceEncoderRepository::getCurrentEncoder ( ) const [virtual]

getCurrentEncoder() implementation of EncoderRepositoryInterface.

See also

EncoderRepositoryInterface::getCurrentEncoder

Implements Coral::EncoderRepositoryInterface.

6.115.3.6 TraceEncoderInterface * Coral::TraceEncoderRepository::getCurrentTraceEncoder ( ) const

Get TraceEncoderInterface of the current encoder.

Returns

Current encoder's TraceEncoderInterface.

6.115.3.7 card32 Coral::TraceEncoderRepository::getFlags ( ) const [virtual]

getFlags() implementation of TraceEncoderInterface.

See also

TraceEncoderInterface::getFlags

Implements Coral::TraceEncoderInterface.

6.115.3.8 double Coral::TraceEncoderRepository::getMaxTransferDelay ( ) const [virtual]

getMaxTransferDelay() implementation of TraceEncoderInterface.

See also

TraceEncoderInterface::getMaxTransferDelay

Implements Coral::TraceEncoderInterface.

6.115.3.9 card64 Coral::TraceEncoderRepository::getMaxWantedBandwidth ( ) const [virtual]

getMaxWantedBandwidth() implementation of TraceEncoderInterface.

See also

TraceEncoderInterface::getMaxWantedBandwidth

Implements Coral::TraceEncoderInterface.

6.115.3.10 card64 Coral::TraceEncoderRepository::getMinWantedBandwidth ( ) const [virtual]

getMinWantedBandwidth() implementation of TraceEncoderInterface.

See also

TraceEncoderInterface::getMinWantedBandwidth

Implements Coral::TraceEncoderInterface.

6.115.3.11 cardinal Coral::TraceEncoderRepository::getNextPacket (EncoderPacket * *encoderPacket*) [virtual]

getNextPacket() implementation of EncoderInterface.

See also

EncoderInterface::getNextPacket

Implements Coral::EncoderInterface.

6.115.3.12 AbstractQoSDescription * Coral::TraceEncoderRepository::getQoSDescription (const cardinal *pktHeaderSize,* const cardinal *pktMaxSize,* const card64 *position*) [virtual]

getQoSDescription() implementation of EncoderInterface.

See also

EncoderInterface::getQoSDescription

Implements Coral::EncoderInterface.

6.115.3.13 int8 Coral::TraceEncoderRepository::getSessionPriority ( ) const [virtual]

getSessionPriority() implementation of TraceEncoderInterface.

See also

TraceEncoderInterface::getSessionPriority

Implements Coral::TraceEncoderInterface.

6.115.3.14 int8 Coral::TraceEncoderRepository::getStreamPriority ( ) const [virtual]

getStreamPriority() implementation of TraceEncoderInterface.

See also

TraceEncoderInterface::getStreamPriority

Implements Coral::TraceEncoderInterface.

6.115.3.15 const card16 Coral::TraceEncoderRepository::getTypeID ( ) const [virtual]

getTypeID() implementation of EncoderInterface.

See also

EncoderInterface::getTypeID

Implements Coral::EncoderInterface.

6.115.3.16 const char * Coral::TraceEncoderRepository::getTypeName ( ) const [virtual]

getTypeName implementation of EncoderInterface.

See also

EncoderInterface::getTypeName

Implements Coral::EncoderInterface.

6.115.3.17 double Coral::TraceEncoderRepository::getWantedUtilization ( ) const [virtual]

getWantedUtilization() implementation of TraceEncoderInterface.

See also

TraceEncoderInterface::getWantedUtilization

Implements Coral::TraceEncoderInterface.

6.115.3.18 bool Coral::TraceEncoderRepository::prepareNextFrame (const cardinal *headerSize*, const cardinal *maxPacketSize*, const cardinal *flags*) [virtual]

prepareNextFrame() implementation of EncoderInterface.

See also

EncoderInterface::prepareNextFrame

Implements Coral::EncoderInterface.

6.115.3.19 void Coral::TraceEncoderRepository::removeEncoder (TraceEncoderInterface * *encoder*)

Remove trace encoder from repository.

Parameters

encoder	Trace encoder to be removed.

6.115.3.20 void Coral::TraceEncoderRepository::reset ( ) [virtual]

reset() implementation of EncoderInterface.

See also

EncoderInterface::reset

Implements Coral::EncoderInterface.

6.115.3.21 bool Coral::TraceEncoderRepository::selectEncoderForTypeID (const card16 *typeID*) [virtual]

selectEncoderForTypeID() implementation of EncoderRepositoryInterface.

See also

EncoderRepositoryInterface::selectEncoderForTypeID

Implements Coral::EncoderRepositoryInterface.

6.115.3.22 void Coral::TraceEncoderRepository::setAutoDelete (const bool *on*) [inline]

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

6.115.3.23 void Coral::TraceEncoderRepository::setFlags (const card32 *flags*) [virtual]

setFlags() implementation of TraceEncoderInterface.

See also

TraceEncoderInterface::setFlags

Implements Coral::TraceEncoderInterface.

6.115.3.24 void Coral::TraceEncoderRepository::setMaxTransferDelay (const double *delay*) [virtual]

setMaxTransferDelay() implementation of TraceEncoderInterface.

See also

TraceEncoderInterface::setMaxWanted

Implements Coral::TraceEncoderInterface.

6.115.3.25 void Coral::TraceEncoderRepository::setMaxWantedBandwidth (const card64 *bandwidth*) [virtual]

setMaxWantedBandwidth() implementation of TraceEncoderInterface.

See also

TraceEncoderInterface::setMaxWantedBandwidth

Implements Coral::TraceEncoderInterface.

6.115.3.26 void Coral::TraceEncoderRepository::setMinWantedBandwidth (const card64 *bandwidth*) [virtual]

setMinWantedBandwidth() implementation of TraceEncoderInterface.

See also

TraceEncoderInterface::setMinWantedBandwidth

Implements Coral::TraceEncoderInterface.

6.115.3.27 void Coral::TraceEncoderRepository::setSessionPriority (const int8 *priority*) [virtual]

setSessionPriority() implementation of TraceEncoderInterface.

See also

TraceEncoderInterface::setSessionPriority

Implements Coral::TraceEncoderInterface.

6.115.3.28 void Coral::TraceEncoderRepository::setStreamPriority (const int8 *priority*) [virtual]

setStreamPriority() implementation of TraceEncoderInterface.

See also

TraceEncoderInterface::setStreamPriority

Implements Coral::TraceEncoderInterface.

6.115.3.29 void Coral::TraceEncoderRepository::setWantedUtilization (const double *utilization*) [virtual]

setWantedUtilization() implementation of TraceEncoderInterface.

See also

TraceEncoderInterface::setWantedUtilization

Implements Coral::TraceEncoderInterface.

6.115.3.30 void Coral::TraceEncoderRepository::updateQuality (const AbstractQoSDescription * *aqd*) [virtual]

updateQuality() implementation of EncoderInterface.

See also

EncoderInterface::updateQuality

Implements Coral::EncoderInterface.

6.115.4 Member Data Documentation

6.115.4.1 bool Coral::TraceEncoderRepository::AutoDelete [private]

6.115.4.2 TraceEncoderInterface* Coral::TraceEncoderRepository::Encoder [private]

6.115.4.3 Coral::TraceEncoderRepository::TraceEncoderRepository [private]

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

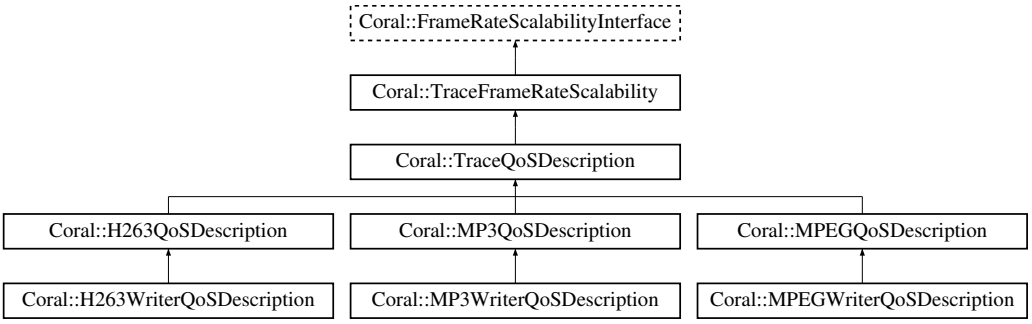
- traceencoderrepository.h
- traceencoderrepository.cc

6.116 Coral::TraceFrameRateScalability Class Reference

Trace Frame Rate Scalability.

`#include <traceframeratescalability.h>`

Inheritance diagram for Coral::TraceFrameRateScalability:



Public Member Functions

- TraceFrameRateScalability ()
- virtual ~TraceFrameRateScalability ()
- void initFrameRateScalability (TDTFReader *traceReader, const card64 position)
- const char * getFrameRateScalabilityClass () const
- bool isFrameRateScalable () const
- double getMinFrameRate () const
- double getMaxFrameRate () const
- bool isValidFrameRate (const double frameRate) const
- double getNearestValidFrameRate (const double frameRate) const
- double getNextFrameRateForRate (const double frameRate) const
- double getPrevFrameRateForRate (const double frameRate) const
- double getFrameRateScaleFactorForRate (const double frameRate) const
- double getFrameRateUtilizationForRate (const double frameRate) const
- double getFrameRateUtilizationWeight (const double frameRate) const

Protected Attributes

- TDTFReader * TraceReader
- card64 Position

6.116.1 Detailed Description

Trace Frame Rate Scalability.

This class is an implementation of FrameRateScalabilityInterface.

Author

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

6.116.2 Constructor & Destructor Documentation

6.116.2.1 Coral::TraceFrameRateScalability::TraceFrameRateScalability ( )

Constructor.

6.116.2.2 Coral::TraceFrameRateScalability::~TraceFrameRateScalability ( ) [virtual]

Destructor.

6.116.3 Member Function Documentation

6.116.3.1 const char * Coral::TraceFrameRateScalability::getFrameRateScalabilityClass ( ) const [virtual]

Implementation of FrameRateScalabilityInterface.

See also

FrameRateScalabilityInterface::getFrameRateScalabilityClass

Implements Coral::FrameRateScalabilityInterface.

Reimplemented in Coral::TraceQoSDescription.

6.116.3.2 double Coral::TraceFrameRateScalability::getFrameRateScaleFactorForRate (const double *frameRate*) const [virtual]

Implementation of FrameRateScalabilityInterface.

See also

FrameRateScalabilityInterface::getFrameRateScaleFactorForRate

Implements Coral::FrameRateScalabilityInterface.

6.116.3.3 double Coral::TraceFrameRateScalability::getFrameRateUtilizationForRate (const double *frameRate*) const [virtual]

Implementation of FrameRateScalabilityInterface.

See also

FrameRateScalabilityInterface::getFrameRateUtilizationForRate

Implements Coral::FrameRateScalabilityInterface.

6.116.3.4 double Coral::TraceFrameRateScalability::getFrameRateUtilizationWeight (const double *frameRate*) const [virtual]

Implementation of FrameRateScalabilityInterface.

See also

FrameRateScalabilityInterface::getFrameRateUtilizationWeight

Implements Coral::FrameRateScalabilityInterface.

6.116.3.5 double Coral::TraceFrameRateScalability::getMaxFrameRate ( ) const [virtual]

Implementation of FrameRateScalabilityInterface.

See also

FrameRateScalabilityInterface::getMaxFrameRate

Implements Coral::FrameRateScalabilityInterface.

6.116.3.6 double Coral::TraceFrameRateScalability::getMinFrameRate ( ) const [virtual]

Implementation of FrameRateScalabilityInterface.

See also

FrameRateScalabilityInterface::getMinFrameRate

Implements Coral::FrameRateScalabilityInterface.

6.116.3.7 double Coral::TraceFrameRateScalability::getNearestValidFrameRate (const double *frameRate*) const [virtual]

Implementation of FrameRateScalabilityInterface.

See also

FrameRateScalabilityInterface::getNearestValidFrameRate

Implements Coral::FrameRateScalabilityInterface.

6.116.3.8 double Coral::TraceFrameRateScalability::getNextFrameRateForRate (const double *frameRate*) const [virtual]

Implementation of FrameRateScalabilityInterface.

See also

FrameRateScalabilityInterface::getNextFrameRateForRate

Implements Coral::FrameRateScalabilityInterface.

6.116.3.9 double Coral::TraceFrameRateScalability::getPrevFrameRateForRate (const double *frameRate*) const [virtual]

Implementation of FrameRateScalabilityInterface.

See also

FrameRateScalabilityInterface::getPrevFrameRateForRate

Implements Coral::FrameRateScalabilityInterface.

6.116.3.10 void Coral::TraceFrameRateScalability::initFrameRateScalability (TDTFReader * *traceReader,* const card64 *position*)

Initialize.

Parameters

traceReader	TDTFReader.
position	Position.

6.116.3.11 bool Coral::TraceFrameRateScalability::isFrameRateScalable ( ) const [virtual]

Implementation of FrameRateScalabilityInterface.

See also

FrameRateScalabilityInterface::isFrameRateScalable

Implements Coral::FrameRateScalabilityInterface.

6.116.3.12 bool Coral::TraceFrameRateScalability::isValidFrameRate (const double *frameRate*) const [virtual]

Implementation of FrameRateScalabilityInterface.

See also

FrameRateScalabilityInterface::isValidFrameRate

Implements Coral::FrameRateScalabilityInterface.

6.116.4 Member Data Documentation

6.116.4.1 card64 Coral::TraceFrameRateScalability::Position [protected]

Reimplemented in Coral::TraceQoSDescription.

6.116.4.2 TDTFReader* Coral::TraceFrameRateScalability::TraceReader [protected]

Reimplemented in Coral::TraceQoSDescription.

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

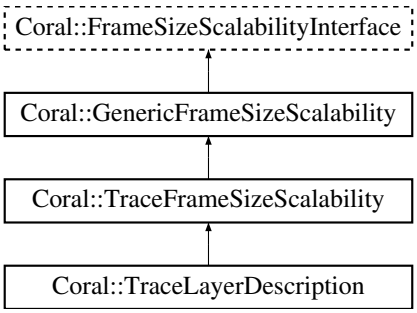
- traceframeratescalability.h
- traceframeratescalability.cc

6.117 Coral::TraceFrameSizeScalability Class Reference

Trace Frame Size Scalability.

`#include <traceframesizescalability.h>`

Inheritance diagram for Coral::TraceFrameSizeScalability:



Public Member Functions

- TraceFrameSizeScalability ()
- ~TraceFrameSizeScalability ()
- void initFrameSizeScalability (TDTFReader *traceReader, const card64 position, const cardinal layer, const double scaleFactor)

- const char * getFrameSizeScalabilityClass () const
- bool isFrameSizeScalable () const
- bool isVariableBitrate () const
- cardinal getMinPayloadFrameSizeForDelay (const double frameRate, const cardinal bufferDelay) const
- cardinal getMaxPayloadFrameSizeForDelay (const double frameRate, const cardinal bufferDelay) const
- cardinal getMaxFrameCountForDelay (const double frameRate, const cardinal bufferDelay) const
- double getPayloadFrameSizeUtilizationForDelayAndSize (const double frameRate, const cardinal bufferDelay, const cardinal frameSize) const
- double getFrameSizeUtilizationWeight (const double frameRate) const
- cardinal getMaxBufferDelay (const double frameRate) const

Protected Attributes

- TDTFReader * TraceReader
- card64 Position
- double ScaleFactor
- cardinal Layer

6.117.1 Detailed Description

Trace Frame Size Scalability.

This class is am implementation of FrameSizeScalabilityInterface. Important node: All frames sizes in this class are payload frame sizes!

Author

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

6.117.2 Constructor & Destructor Documentation

6.117.2.1 Coral::TraceFrameSizeScalability::TraceFrameSizeScalability ( )

Constructor.

6.117.2.2 Coral::TraceFrameSizeScalability::~TraceFrameSizeScalability ( )

Desstructor.

6.117.3 Member Function Documentation

6.117.3.1 const char * Coral::TraceFrameSizeScalability::getFrameSizeScalabilityClass ( ) const [virtual]

Implementation of FrameSizeScalabilityInterface.

See also

FrameSizeScalabilityInterface::getFrameSizeScalabilityClass

Implements Coral::FrameSizeScalabilityInterface.

Reimplemented in Coral::TraceLayerDescription.

6.117.3.2 double Coral::TraceFrameSizeScalability::getFrameSizeUtilizationWeight (const double *frameRate*) const [virtual]

Implementation of FrameSizeScalabilityInterface.

See also

FrameSizeScalabilityInterface::getFrameSizeUtilizationWeight

Reimplemented from Coral::GenericFrameSizeScalability.

6.117.3.3 cardinal Coral::TraceFrameSizeScalability::getMaxBufferDelay (const double *frameRate*) const [virtual]

Implementation of FrameSizeScalabilityInterface.

See also

FrameSizeScalabilityInterface::getMaxBufferDelay

Implements Coral::FrameSizeScalabilityInterface.

6.117.3.4 cardinal Coral::TraceFrameSizeScalability::getMaxFrameCountForDelay (const double *frameRate,* const cardinal *bufferDelay*) const [virtual]

Implementation of FrameSizeScalabilityInterface.

See also

FrameSizeScalabilityInterface::getMaxFrameCountForDelay

Implements Coral::FrameSizeScalabilityInterface.

6.117.3.5 cardinal Coral::TraceFrameSizeScalability::getMaxPayloadFrameSizeForDelay (const double *frameRate,* const cardinal *bufferDelay*) const [virtual]

Implementation of FrameSizeScalabilityInterface.

See also

FrameSizeScalabilityInterface::getMaxPayloadFrameSizeForDelay

Implements Coral::FrameSizeScalabilityInterface.

6.117.3.6 cardinal Coral::TraceFrameSizeScalability::getMinPayloadFrameSizeForDelay (const double *frameRate,* const cardinal *bufferDelay*) const [virtual]

Implementation of FrameSizeScalabilityInterface.

See also

FrameSizeScalabilityInterface::getMinPayloadFrameSizeForDelay

Implements Coral::FrameSizeScalabilityInterface.

6.117.3.7 double Coral::TraceFrameSizeScalability::getPayloadFrameSizeUtilizationForDelayAndSize (const double *frameRate,* const cardinal *bufferDelay,* const cardinal *frameSize*) const [virtual]

Implementation of FrameSizeScalabilityInterface.

See also

FrameSizeScalabilityInterface::getPayloadFrameSizeUtilizationForDelayAndSize

Reimplemented from Coral::GenericFrameSizeScalability.

6.117.3.8 void Coral::TraceFrameSizeScalability::initFrameSizeScalability (TDTFReader * *traceReader,* const card64 *position,* const cardinal *layer,* const double *scaleFactor*)

Initialize object with new maximum payload frame size and scale factor. MinFrameSize = scaleFactor * MaxFrameSize.

Parameters

traceReader	TDTFReader.
position	Position (for TDTFReader).
layer	Layer number (for TDTFReader).
scaleFactor	Scale factor.

6.117.3.9 bool Coral::TraceFrameSizeScalability::isFrameSizeScalable ( ) const [virtual]

Implementation of FrameSizeScalabilityInterface.

See also

FrameSizeScalabilityInterface::isFrameSizeScalable

Implements Coral::FrameSizeScalabilityInterface.

6.117.3.10 bool Coral::TraceFrameSizeScalability::isVariableBitrate ( ) const [virtual]

Implementation of FrameSizeScalabilityInterface.

See also

FrameSizeScalabilityInterface::isVariableBitRate

Implements Coral::FrameSizeScalabilityInterface.

6.117.4 Member Data Documentation

6.117.4.1 cardinal Coral::TraceFrameSizeScalability::Layer [protected]

6.117.4.2 card64 Coral::TraceFrameSizeScalability::Position [protected]

6.117.4.3 double Coral::TraceFrameSizeScalability::ScaleFactor [protected]

6.117.4.4 TDTFReader* Coral::TraceFrameSizeScalability::TraceReader [protected]

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

- traceframesizescalability.h
- traceframesizescalability.cc

6.118 Coral::TraceHeader Struct Reference

Trace Header.

`#include <tdtf.h>`

Public Attributes

- card64 FrameRate
- card32 Frames
- card8 Layers
- card8 pad01
- card16 pad02
- FrameDescription Frame [0]

6.118.1 Detailed Description

Trace Header.

This is the header for a trace.

Author

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

6.118.2 Member Data Documentation

6.118.2.1 FrameDescription Coral::TraceHeader::Frame[0]

Frame description array: 0 .. Frames - 1 Layer #0 trace Frames .. 2 * Frames - 1 Layer #1 trace ...

6.118.2.2 card64 Coral::TraceHeader::FrameRate

Frame rate (double converted to binary using translateToBinary()).

See also

translateToBinary
translateToDouble

6.118.2.3 card32 Coral::TraceHeader::Frames

Number of frames.

6.118.2.4 card8 Coral::TraceHeader::Layers

Number of layers.

6.118.2.5 card8 Coral::TraceHeader::pad01

Unused. Should be set to 0.

6.118.2.6 card16 Coral::TraceHeader::pad02

Unused. Should be set to 0.

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

- tdtf.h

6.119 Coral::TraceLayerConfiguration Struct Reference

Trace Layer Configuration.

`#include <traceconfiguration.h>`

Public Attributes

- double Scalability
- double CostFactor
- cardinal ByterateEmpiricalEnvelopePairs
- cardinal FrameCountEmpiricalEnvelopePairs
- cardinal LayerFlags
- card16 FrameSizeUtilizationMaxConstants
- card16 FrameSizeUtilizationConstants
- card16 FrameSizeUtilizationType
- double FrameSizeUtilizationWeight
- double FrameSizeUtilizationConstant [MaxFrameSizeUtilizationConstants]

Static Public Attributes

- static const card16 MaxFrameSizeUtilizationConstants = 8

6.119.1 Detailed Description

Trace Layer Configuration.

This is a layer's trace configuration.

Author

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

6.119.2 Member Data Documentation

6.119.2.1 cardinal Coral::TraceLayerConfiguration::ByterateEmpiricalEnvelopePairs

Number of pairs in byterate empirical envelope.

6.119.2.2 double Coral::TraceLayerConfiguration::CostFactor

Cost factor for this layer.

6.119.2.3 cardinal Coral::TraceLayerConfiguration::FrameCountEmpiricalEnvelopePairs

Number of pairs in frame count empirical envelope.

6.119.2.4 double Coral::TraceLayerConfiguration::FrameSizeUtilizationConstant[MaxFrameSizeUtilizationConstants]

Array of frame size utilization constants.

6.119.2.5 card16 Coral::TraceLayerConfiguration::FrameSizeUtilizationConstants

Number of frame size utilization constants.

6.119.2.6 card16 Coral::TraceLayerConfiguration::FrameSizeUtilizationMaxConstants

Number of frame size utilization constants.

6.119.2.7 card16 Coral::TraceLayerConfiguration::FrameSizeUtilizationType

Frame size utility function type.

6.119.2.8 double Coral::TraceLayerConfiguration::FrameSizeUtilizationWeight

Frame size utilization weight.

6.119.2.9 cardinal Coral::TraceLayerConfiguration::LayerFlags

Layer flags.

6.119.2.10 const card16 Coral::TraceLayerConfiguration::MaxFrameSize-UtilizationConstants = 8 [static]

Maximum number of frame size utilization constants.

6.119.2.11 double Coral::TraceLayerConfiguration::Scalability

Scalability. MinFrameSize = Scalability ∗ MaxFrameSize.

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

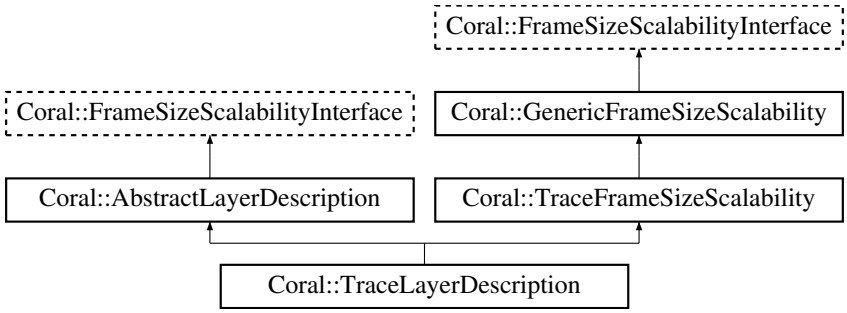
- traceconfiguration.h

6.120 Coral::TraceLayerDescription Class Reference

Trace Layer QoS Description.

`#include <traceqosdescription.h>`

Inheritance diagram for Coral::TraceLayerDescription:



Public Member Functions

- const char ∗ getFrameSizeScalabilityClass () const

6.120.1 Detailed Description

Trace Layer QoS Description.

This is the QoS description of a trace layer.

Author

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

6.120.2 Member Function Documentation

6.120.2.1 const char * Coral::TraceLayerDescription::getFrameSizeScalabilityClass ( ) const [virtual]

Reimplementation of GenericVariableBitrateFrameSizeScalability's getFrameSizeScalabilityClass().

See also

GenericVariableBitrateFrameSizeScalability::getFrameSizeScalabilityClass
FrameSizeScalabilityInterface::getFrameSizeScalabilityClass

Reimplemented from Coral::TraceFrameSizeScalability.

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

- traceqosdescription.h
- traceqosdescription.cc

6.121 Coral::TracePacket Class Reference

Trace Packet.

`#include <tracepacket.h>`

Public Types

- enum TraceFlags { TF_None = 0, TF_Information = 1 }

Public Member Functions

- TracePacket ()
- void translate ()
- void reset ()

Public Attributes

- card32 FormatID
- card8 Layer
- card8 Layers
- card8 ErrorCode
- card8 Flags
- card64 Position
- card64 MaxPosition
- card32 FrameID
- card32 Offset
- char Data [0]

Static Public Attributes

- static const card16 TraceTypeID = 0x8833
- static const char TraceTypeName [] = "Trace Encoding"
- static const card32 TraceFormatID = 0x33140000 | TraceTypeID

6.121.1 Detailed Description

Trace Packet.

This class defines the packet format for the trace encoder.

Author

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

See also

TraceEncoder
TraceDecoder

6.121.2 Member Enumeration Documentation

6.121.2.1 enum Coral::TracePacket::TraceFlags

Emumeration of Flags.

Enumerator:

TF_None

TF_Information

6.121.3 Constructor & Destructor Documentation

6.121.3.1 Coral::TracePacket::TracePacket ( )

Constructor.

6.121.4 Member Function Documentation

6.121.4.1 void Coral::TracePacket::reset ( )

Reset report.

6.121.4.2 void Coral::TracePacket::translate ( )

Translate byte order.

6.121.5 Member Data Documentation

6.121.5.1 char Coral::TracePacket::Data[0]

Packet data.

6.121.5.2 card8 Coral::TracePacket::ErrorCode

Error code.

6.121.5.3 card8 Coral::TracePacket::Flags

Flags.

6.121.5.4 card32 Coral::TracePacket::FormatID

Packet format ID.

6.121.5.5 card32 Coral::TracePacket::FrameID

Frame ID.

6.121.5.6 card8 Coral::TracePacket::Layer

Layer number

6.121.5.7 card8 Coral::TracePacket::Layers

Number of layers

6.121.5.8 card64 Coral::TracePacket::MaxPosition

Maximum position in nanoseconds.

6.121.5.9 card32 Coral::TracePacket::Offset

Fragment offset.

6.121.5.10 card64 Coral::TracePacket::Position

Current position in nanoseconds.

6.121.5.11 const card32 Coral::TracePacket::TraceFormatID = 0x33140000 | TraceTypeID [static]

Trace Encoding package format ID.

6.121.5.12 const card16 Coral::TracePacket::TraceTypeID = 0x8833 [static]

Type ID for Trace Encoding.

6.121.5.13 const char Coral::TracePacket::TraceTypeName = "Trace Encoding" [static]

Name for Trace Encoding.

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

- tracepacket.h
- tracepacket.cc

6.122 Coral::TracePacketData Struct Reference

Trace Packet Data.

`#include <tracepacket.h>`

Public Attributes

- card64 FrameRate
- card64 Utilization
- card64 Bandwidth
- card64 MinBandwidth
- card64 MaxBandwidth
- MediaInfo Information
- card32 Flags
- int8 StreamPriority
- int8 SessionPriority

6.122.1 Detailed Description

Trace Packet Data.

This is the trace packet payload.

Author

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

6.122.2 Member Data Documentation

6.122.2.1 card64 Coral::TracePacketData::Bandwidth

Bandwidth.

6.122.2.2 card32 Coral::TracePacketData::Flags

Flags.

6.122.2.3 card64 Coral::TracePacketData::FrameRate

Frame rate.

6.122.2.4 MediaInfo Coral::TracePacketData::Information

Media Info.

6.122.2.5 card64 Coral::TracePacketData::MaxBandwidth

Maximum bandwidth.

6.122.2.6 card64 Coral::TracePacketData::MinBandwidth

Minimum bandwidth.

6.122.2.7 int8 Coral::TracePacketData::SessionPriority

Session priority.

6.122.2.8 int8 Coral::TracePacketData::StreamPriority

Stream priority.

6.122.2.9 card64 Coral::TracePacketData::Utilization

Utilization.

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

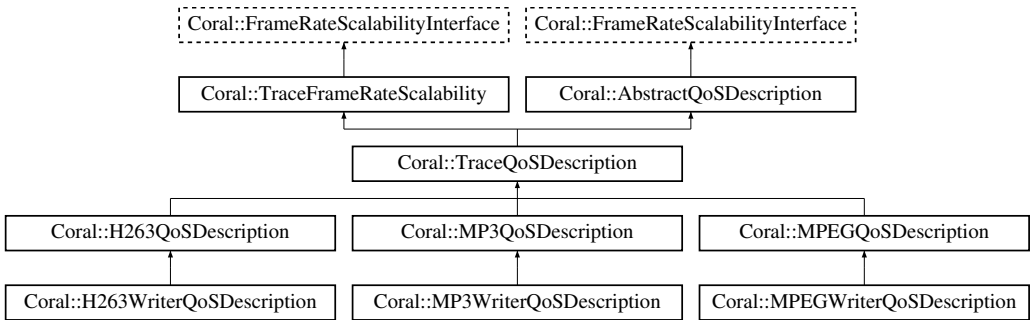
- tracepacket.h

6.123 Coral::TraceQoSDescription Class Reference

Trace QoS Description.

`#include <traceqosdescription.h>`

Inheritance diagram for Coral::TraceQoSDescription:



Public Member Functions

- void initTraceDescription (TDTFReader *traceReader, const card64 position, const card64 maxPosition, const double frameRate)
- void updateDescription (const cardinal pktHeaderSize, const cardinal pktMaxSize)
- cardinal getLayers () const
- AbstractLayerDescription * getLayer (const cardinal layer) const
- const char * getFrameRateScalabilityClass () const
- cardinal getPrecomputedResourceUtilizationList (ResourceUtilizationPoint *rup, const card64 bwThreshold, const double utThreshold, const cardinal maxPoints) const
- double calculateUtilizationForLayerBandwidths (const double frameRate, const cardinal layers, const card64 *bandwidth) const

Static Public Attributes

- static const cardinal MaxLayers = 16

Protected Member Functions

- void doSelectIteration (bool *select, 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

Protected Attributes

- TDTFReader * TraceReader
- card64 Position
- card64 MaxPosition
- double FrameRate
- TraceLayerDescription Layer [MaxLayers]

6.123.1 Detailed Description

Trace QoS Description.

This is the QoS description of a trace stream.

Author

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

6.123.2 Member Function Documentation

6.123.2.1 double Coral::TraceQoSDescription::calculateUtilizationForLayerBandwidths (const double *frameRate,* const cardinal *layers,* const card64 * *bandwidth*) const [virtual]

Implementation of AbstractQoSDescription's calculateUtilizationForLayerBandwidths().

See also

AbstractQoSDescription::calculateUtilizationForLayerBandwidths

Reimplemented from Coral::AbstractQoSDescription.

6.123.2.2 void Coral::TraceQoSDescription::doSelectIteration (bool * *select,* 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 [protected]

6.123.2.3 const char * Coral::TraceQoSDescription::getFrameRateScalabilityClass ( ) const [virtual]

Reimplementation of GenericFrameRateScalability's getFrameRateScalabilityClass().

See also

GenericFrameRateScalability::getFrameRateScalabilityClass
FrameRateScalabilityInterface::getFrameRateScalabilityClass

Reimplemented from Coral::TraceFrameRateScalability.

6.123.2.4 AbstractLayerDescription * Coral::TraceQoSDescription::getLayer (const cardinal *layer*) const [virtual]

Implementation of AbstractQoSDescription's getLayer().

See also

AbstractQoSDescription::getLayer

Implements Coral::AbstractQoSDescription.

6.123.2.5 cardinal Coral::TraceQoSDescription::getLayers ( ) const [virtual]

Implementation of AbstractQoSDescription's getLayers().

See also

AbstractQoSDescription::getLayers

Implements Coral::AbstractQoSDescription.

6.123.2.6 cardinal Coral::TraceQoSDescription::getPrecomputedResourceUtilizationList (ResourceUtilizationPoint * *rup*, const card64 *bwThreshold*, const double *utThreshold*, const cardinal *maxPoints*) const [virtual]

Implementation of AbstractQoSDescription's getPrecomputedResourceUtilizationList().

See also

AbstractQoSDescription::getPrecomputedResourceUtilizationList

Implements Coral::AbstractQoSDescription.

6.123.2.7 void Coral::TraceQoSDescription::initTraceDescription (TDTFReader * *traceReader*, const card64 *position*, const card64 *maxPosition*, const double *frameRate*)

Initialize description.

Parameters

traceReader	TDTFReader.
position	RTP Position.
position	Maximum RTP Position.
frameRate	Frame rate.

6.123.2.8 void Coral::TraceQoSDescription::updateDescription (const cardinal *pktHeaderSize*, const cardinal *pktMaxSize*) [virtual]

Implementation of AbstractQoSDescription's updateDescription().

See also

AbstractQoSDescription::updateDescription

Implements Coral::AbstractQoSDescription.

6.123.3 Member Data Documentation

6.123.3.1 double Coral::TraceQoSDescription::FrameRate [protected]

Reimplemented from Coral::AbstractQoSDescription.

6.123.3.2 TraceLayerDescription Coral::TraceQoSDescription::Layer[MaxLayers] [protected]

6.123.3.3 const cardinal Coral::TraceQoSDescription::MaxLayers = 16 [static]

Maximum number of layers.

6.123.3.4 card64 Coral::TraceQoSDescription::MaxPosition [protected]

6.123.3.5 card64 Coral::TraceQoSDescription::Position [protected]

Reimplemented from Coral::AbstractQoSDescription.

6.123.3.6 TDTFReader* Coral::TraceQoSDescription::TraceReader [protected]

Reimplemented from Coral::TraceFrameRateScalability.

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

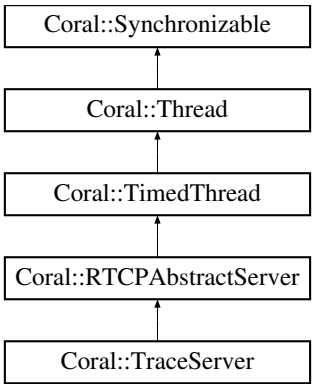
- traceqosdescription.h
- traceqosdescription.cc

6.124 Coral::TraceServer Class Reference

Trace Server.

`#include <traceserver.h>`

Inheritance diagram for Coral::TraceServer:



Classes

- struct User

Public Member Functions

- TraceServer (BandwidthManager *bwManager, const cardinal maxPacketSize=1500)
- ~TraceServer ()
- card32 getOurSSRC () const
- bool getLossScalability () const
- void setLossScalability (const bool on)
- cardinal getMaxPacketSize () const
- cardinal setMaxPacketSize (const cardinal size)
- void outOfMemoryWarning ()
- void * newClient (Client *client, const char *cname)
- void deleteClient (Client *client, const DeleteReason reason)
- bool checkClient (const Client *client)
- void appMessage (const Client *client, const char *name, const void *data, const cardinal dataLength)
- void sdesMessage (const Client *client, const card8 type, const char *data, const cardinal length)
- void receiverReport (const Client *client, const RTCPReceptionReportBlock *report, const cardinal layer)
- void userCommand (const Client *client, User *user, const TraceClientAppPacket *app)
- void managementUpdate (const Client *client, User *user)

Private Attributes

- BandwidthManager * BandwidthMgr
- multimap< const cardinal, User * > UserSet
- Synchronizable UserSetSync
- cardinal MaxPacketSize
- card32 OurSSRC
- bool LossScalability

6.124.1 Detailed Description

Trace Server.

This class is an trace server based on RTCPAbstractServer

Author

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

6.124.2 Constructor & Destructor Documentation

6.124.2.1 Coral::TraceServer::TraceServer (BandwidthManager * *bwManager,* const cardinal *maxPacketSize* = 1500)

Constructor for new TraceServer.

Parameters

bwManager	BandwidthManager object.
maxPacket-Size	Maximum packet size.

6.124.2.2 Coral::TraceServer::~TraceServer ( )

Destructor.

6.124.3 Member Function Documentation

6.124.3.1 void Coral::TraceServer::appMessage (const Client * *client,* const char * *name,* const void * *data,* const cardinal *dataLength*) [virtual]

appMessage() implementation of RTCPAbstractServer.

See also

RTCPAbstractServer::appMessage

Implements Coral::RTCPAbstractServer.

6.124.3.2 bool Coral::TraceServer::checkClient (const Client * *client*) [virtual]

checkClient() implementation of RTCPAbstractServer.

See also

RTCPAbstractServer::checkClient

Implements Coral::RTCPAbstractServer.

6.124.3.3 void Coral::TraceServer::deleteClient (Client * *client,* const DeleteReason *reason*) [virtual]

deleteClient() implementation of RTCPAbstractServer.

See also

RTCPAbstractServer::deleteClient

Implements Coral::RTCPAbstractServer.

6.124.3.4 bool Coral::TraceServer::getLossScalability ( ) const [inline]

Get loss scalibility setting.

Returns

true, if loss scalability is on; false otherwise.

6.124.3.5 cardinal Coral::TraceServer::getMaxPacketSize ( ) const [inline]

Get maximum packet size.

Returns

Maximum packet size.

6.124.3.6 card32 Coral::TraceServer::getOurSSRC ( ) const

Get client SSRC.

Returns

Client SSRC.

6.124.3.7 void Coral::TraceServer::managementUpdate (const Client * *client*, User * *user*)

Update QoS/congestion management.

Parameters

client	Client to do congestion for.
user	User data.

6.124.3.8 void * Coral::TraceServer::newClient (Client * *client*, const char * *cname*) [virtual]

newClient() implementation of RTCPAbstractServer.

See also

RTCPAbstractServer::newClient

Implements Coral::RTCPAbstractServer.

6.124.3.9 void Coral::TraceServer::outOfMemoryWarning ( ) [virtual]

outOfMemoryWarning() implementation of RTCPAbstractServer.

Reimplemented from Coral::RTCPAbstractServer.

6.124.3.10 void Coral::TraceServer::receiverReport (const Client * *client*, const RTCPReceptionReportBlock * *report*, const cardinal *layer*) [virtual]

receiverReport() implementation of RTCPAbstractServer.

See also

RTCPAbstractServer::receiverReport

Implements Coral::RTCPAbstractServer.

6.124.3.11 void Coral::TraceServer::sdesMessage (const Client * *client*, const card8 *type*, const char * *data*, const cardinal *length*) [virtual]

sdesMessage() implementation of RTCPAbstractServer.

See also

RTCPAbstractServer::sdesMessage

Implements Coral::RTCPAbstractServer.

6.124.3.12 void Coral::TraceServer::setLossScalability (const bool *on*) [inline]

Set loss scalibility setting.

Parameters

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

6.124.3.13 cardinal Coral::TraceServer::setMaxPacketSize (const cardinal *size*) [inline]

Set maximum packet size.

Parameters

size	Maximum packet size.

Returns

Maximum packet size set.

6.124.3.14 void Coral::TraceServer::userCommand (const Client * *client*, User * *user*, const TraceClientAppPacket * *app*)

Execute commands given in TraceClientAppPacket.

Parameters

client	Client.
user	User.
app	TraceClientApp message.

6.124.4 Member Data Documentation

6.124.4.1 BandwidthManager* Coral::TraceServer::BandwidthMgr [private]

6.124.4.2 bool Coral::TraceServer::LossScalability [private]

6.124.4.3 cardinal Coral::TraceServer::MaxPacketSize [private]

6.124.4.4 card32 Coral::TraceServer::OurSSRC [private]

6.124.4.5 multimap<const cardinal,User*> Coral::TraceServer::UserSet [private]

6.124.4.6 Synchronizable Coral::TraceServer::UserSetSync [private]

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

- traceserver.h
- traceserver.cc

6.125 Coral::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]

6.125.1 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

6.125.2 Member Function Documentation

6.125.2.1 cardinal Coral::TrafficClassValues::getIndexForTrafficClass (const card8 *trafficClass*) [static]

Get index for given traffic class.

Parameters

trafficClass	Traffic class.

Returns

Index.

6.125.2.2 static const char* Coral::TrafficClassValues::getNameForIndex (const cardinal *index*) [inline, static]

Get name for index entry.

Parameters

index	Index.

Returns

Name.

6.125.2.3 const char * Coral::TrafficClassValues::getNameForTrafficClass (const card8 *trafficClass*) [static]

Get name for given traffic class.

Parameters

trafficClass	Traffic class.

Returns

Name.

6.125.2.4 static card8 Coral::TrafficClassValues::getTrafficClassForIndex (const cardinal *index*) [inline, static]

Get traffic class of given index.

Parameters

index	Index.

Returns

Traffic class.

6.125.2.5 const card16 Coral::TrafficClassValues::getTrafficClassForName (const char * *name*) [static]

Get traffic class for name.

Parameters

name	Name.

Returns

Traffic class or 0xffff, if name in unknown.

6.125.3 Member Data Documentation

6.125.3.1 const cardinal Coral::TrafficClassValues::MaxValues = 16 [static]

Number of values.

6.125.3.2 const char * Coral::TrafficClassValues::TCNames [static, private]

Initial value:

```
{
  "EF",
  "AF11","AF12","AF13",
  "AF21","AF22","AF23",
  "AF31","AF32","AF33",
  "AF41","AF42","AF43",
  "TD1", "TD2",
  "BE"
}
```

6.125.3.3 const card8 Coral::TrafficClassValues::TCValues [static, private]

Initial value:

```
{
  46,10,12,14,18,20,22,26,28,30,34,36,38,40,42,0
}
```

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

- trafficclassvalues.h
- trafficclassvalues.cc

6.126 Coral::TrafficPolicer Class Reference

Traffic Policer.

`#include <trafficpolicer.h>`

Classes

- struct TrafficPolicerPacket

Public Member Functions

- TrafficPolicer ()
- ~TrafficPolicer ()
- card64 getBandwidth () const
- void setBandwidth (const card64 bandwidth)
- double getBufferDelay () const
- void setBufferDelay (const double bufferDelay)
- void setFlush (const bool on)
- void flush ()
- bool refreshBuffer (const card8 trafficClass, const bool doRemapping)
- cardinal write (const cardinal bytes, const card8 trafficClass, const cardinal headerSize)
- void timerEvent ()

Static Public Attributes

- static card64 SimulatorTime = 0

Private Attributes

- card64 SendTimeStamp
- card64 Bandwidth
- double BufferDelay
- bool ExceedFlush
- multimap< card64, TrafficPolicerPacket * > PacketSet

6.126.1 Detailed Description

Traffic Policer.

This class is a traffic policer.

Author

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

6.126.2 Constructor & Destructor Documentation

6.126.2.1 Coral::TrafficPolicer::TrafficPolicer ( )

Constructor.

6.126.2.2 Coral::TrafficPolicer::~TrafficPolicer ( )

Destructor.

6.126.3 Member Function Documentation

6.126.3.1 void Coral::TrafficPolicer::flush ( )

Flush buffer.

6.126.3.2 card64 Coral::TrafficPolicer::getBandwidth ( ) const [inline]

Get bandwidth for following packets.

Returns

Bandwidth.

6.126.3.3 double Coral::TrafficPolicer::getBufferDelay ( ) const [inline]

Get maximum buffer delay for following packets.

Returns

Maximum buffer delay in microseconds.

6.126.3.4 bool Coral::TrafficPolicer::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.

6.126.3.5 void Coral::TrafficPolicer::setBandwidth (const card64 *bandwidth*) [inline]

Set bandwidth for following packets.

Parameters

bandwidth	Bandwidth.

6.126.3.6 void Coral::TrafficPolicer::setBufferDelay (const double *bufferDelay*) [inline]

Set maximum buffer delay for following packets.

Parameters

maxDelay	Maximum buffer delay in microseconds.

6.126.3.7 void Coral::TrafficPolicer::setFlush (const bool *on*) [inline]

Enable ∗one∗ buffer flush, if limits are exceeded.

Parameters

on	true, to enable one buffer flush; false otherwise.

6.126.3.8 void Coral::TrafficPolicer::timerEvent ( )

Implementation of TimedThread's timerEvent() method.

See also

TimedThread::timerEvent

6.126.3.9 cardinal Coral::TrafficPolicer::write (const cardinal *bytes*, const card8 *trafficClass*, const cardinal *headerSize*)

Write given number of bytes into buffer.

Parameters

bytes	Number of bytes.
trafficClass	Traffic class for error output.
headerSize	Packet header size (e.g. 48 for UDP/IPv6).

Returns

0, if write was successful; 1, if limits have been exceeded but allowed by setFlush(); >1, if limits have been exceeded unallowed.

6.126.4 Member Data Documentation

6.126.4.1 card64 Coral::TrafficPolicer::Bandwidth [private]

6.126.4.2 double Coral::TrafficPolicer::BufferDelay [private]

6.126.4.3 bool Coral::TrafficPolicer::ExceedFlush [private]

6.126.4.4 multimap<card64,TrafficPolicerPacket*> Coral::TrafficPolicer::PacketSet [private]

6.126.4.5 card64 Coral::TrafficPolicer::SendTimeStamp [private]

6.126.4.6 card64 Coral::TrafficPolicer::SimulatorTime = 0 [static]

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

- trafficpolicer.h
- trafficpolicer.cc

6.127 Coral::TrafficPolicer::TrafficPolicerPacket Struct Reference

Public Attributes

- card64 SendTimeStamp
- card64 Bandwidth
- double BufferDelay
- cardinal HeaderSize
- cardinal PayloadSize
- card8 TrafficClass
- char Data [0]

6.127.1 Member Data Documentation

6.127.1.1 card64 Coral::TrafficPolicer::TrafficPolicerPacket::Bandwidth

6.127.1.2 double Coral::TrafficPolicer::TrafficPolicerPacket::BufferDelay

6.127.1.3 char Coral::TrafficPolicer::TrafficPolicerPacket::Data[0]

6.127.1.4 cardinal Coral::TrafficPolicer::TrafficPolicerPacket::HeaderSize

6.127.1.5 cardinal Coral::TrafficPolicer::TrafficPolicerPacket::PayloadSize

6.127.1.6 card64 Coral::TrafficPolicer::TrafficPolicerPacket::SendTimeStamp

6.127.1.7 card8 Coral::TrafficPolicer::TrafficPolicerPacket::TrafficClass

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

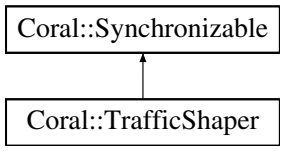
- trafficpolicer.h

6.128 Coral::TrafficShaper Class Reference

Traffic Shaper.

`#include <trafficshaper.h>`

Inheritance diagram for Coral::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 () const
- 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

- deque< TrafficShaperPacket > Queue
- Socket * SenderSocket
- card64 SendTimeStamp
- card64 Bandwidth
- double BufferDelay
- integer LastError
- cardinal LastSeqNum

Static Private Attributes

- static TrafficShaperSingleton Singleton

Friends

- class TrafficShaperSingleton

6.128.1 Detailed Description

Traffic Shaper.

This class is a traffic shaper.

Author

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

6.128.2 Member Enumeration Documentation

6.128.2.1 enum Coral::TrafficShaper::TrafficShaperCommand [private]

Enumerator:

TSC_Write

TSC_Send

TSC_SendTo

6.128.3 Constructor & Destructor Documentation

6.128.3.1 Coral::TrafficShaper::TrafficShaper ( )

Constructor.

6.128.3.2 Coral::TrafficShaper::TrafficShaper (Socket * *socket*)

Constructor.

6.128.3.3 Coral::TrafficShaper::~TrafficShaper ( )

Destructor.

6.128.4 Member Function Documentation

6.128.4.1 ssize_t Coral::TrafficShaper::addPacket (const void * *data*, const cardinal *bytes*, const cardinal *seqNum*, InternetFlow & *destination*, const cardinal *flags*, const cardinal *command*) [private]

6.128.4.2 void Coral::TrafficShaper::flush ( )

Flush buffer.

6.128.4.3 card64 Coral::TrafficShaper::getBandwidth ( ) const [inline]

Get bandwidth for following packets.

Returns

Bandwidth.

6.128.4.4 double Coral::TrafficShaper::getBufferDelay ( ) const [inline]

Get maximum buffer delay for following packets.

Returns

Maximum buffer delay in microseconds.

6.128.4.5 cardinal Coral::TrafficShaper::getLastSeqNum ( ) const [inline]

Get sequence number of last packet sent.

Returns

Sequence number.

6.128.4.6 void Coral::TrafficShaper::init (Socket * *socket*)

Initialize.

Parameters

socket	Socket.

6.128.4.7 bool Coral::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.

6.128.4.8 ssize_t Coral::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.

6.128.4.9 void Coral::TrafficShaper::sendAll ( ) [private]

6.128.4.10 ssize_t Coral::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.

6.128.4.11 void Coral::TrafficShaper::setBandwidth (const card64 *bandwidth*) [inline]

Set bandwidth for following packets.

Parameters

bandwidth	Bandwidth.

6.128.4.12 void Coral::TrafficShaper::setBufferDelay (const double *bufferDelay*) [inline]

Set maximum buffer delay for following packets.

Parameters

bufferDelay	Maximum buffer delay in microseconds.

6.128.4.13 void Coral::TrafficShaper::setSocket (Socket * *socket*) [inline]

Set socket to send shaped traffic to.

Parameters

socket	Socket.

6.128.4.14 ssize_t Coral::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.

6.128.5 Friends And Related Function Documentation

6.128.5.1 friend class TrafficShaperSingleton [friend]

6.128.6 Member Data Documentation

6.128.6.1 card64 Coral::TrafficShaper::Bandwidth [private]

6.128.6.2 double Coral::TrafficShaper::BufferDelay [private]

6.128.6.3 integer Coral::TrafficShaper::LastError [private]

6.128.6.4 cardinal Coral::TrafficShaper::LastSeqNum [private]

6.128.6.5 deque<TrafficShaperPacket> Coral::TrafficShaper::Queue [private]

6.128.6.6 Socket* Coral::TrafficShaper::SenderSocket [private]

6.128.6.7 card64 Coral::TrafficShaper::SendTimeStamp [private]

6.128.6.8 TrafficShaperSingleton Coral::TrafficShaper::Singleton [static, private]

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

- trafficshaper.h
- trafficshaper.cc

6.129 Coral::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

6.129.1 Member Function Documentation

6.129.1.1 int Coral::TrafficShaper::TrafficShaperPacket::operator< (const TrafficShaperPacket & *packet*) const [inline]

6.129.2 Member Data Documentation

6.129.2.1 cardinal Coral::TrafficShaper::TrafficShaperPacket::Command

6.129.2.2 char* Coral::TrafficShaper::TrafficShaperPacket::Data

6.129.2.3 InternetFlow Coral::TrafficShaper::TrafficShaperPacket::Destination

6.129.2.4 cardinal Coral::TrafficShaper::TrafficShaperPacket::Flags

6.129.2.5 cardinal Coral::TrafficShaper::TrafficShaperPacket::HeaderSize

6.129.2.6 cardinal Coral::TrafficShaper::TrafficShaperPacket::PayloadSize

6.129.2.7 card64 Coral::TrafficShaper::TrafficShaperPacket::SendTimeStamp

6.129.2.8 cardinal Coral::TrafficShaper::TrafficShaperPacket::SeqNum

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

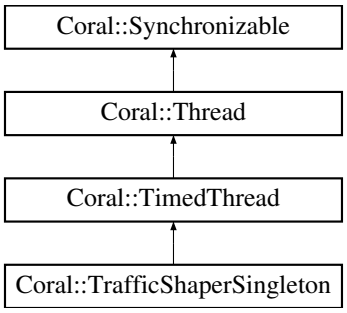
- trafficshaper.h

6.130 Coral::TrafficShaperSingleton Class Reference

Traffic Shaper Singleton.

`#include <trafficshaper.h>`

Inheritance diagram for Coral::TrafficShaperSingleton:



Public Member Functions

- TrafficShaperSingleton ()
- ~TrafficShaperSingleton ()
- void addTrafficShaper (TrafficShaper *ts)
- void removeTrafficShaper (TrafficShaper *ts)

Private Member Functions

- void timerEvent ()

Private Attributes

- vector< TrafficShaper * > ShaperSet
- cardinal UserCount

6.130.1 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

6.130.2 Constructor & Destructor Documentation

6.130.2.1 Coral::TrafficShaperSingleton::TrafficShaperSingleton ( )

Constructor.

6.130.2.2 Coral::TrafficShaperSingleton::~TrafficShaperSingleton ( )

Destructor.

6.130.3 Member Function Documentation

6.130.3.1 void Coral::TrafficShaperSingleton::addTrafficShaper (TrafficShaper * *ts*)

Add traffic shaper object.

Parameters

ts	TrafficShaper.

6.130.3.2 void Coral::TrafficShaperSingleton::removeTrafficShaper (TrafficShaper * *ts*)

Remove traffic shaper object.

Parameters

ts	TrafficShaper.

6.130.3.3 void Coral::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 Coral::TimedThread.

6.130.4 Member Data Documentation

6.130.4.1 vector<TrafficShaper*> Coral::TrafficShaperSingleton::ShaperSet [private]

6.130.4.2 cardinal Coral::TrafficShaperSingleton::UserCount [private]

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

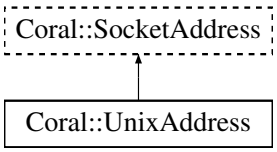
- trafficshaper.h
- trafficshaper.cc

6.131 Coral::UnixAddress Class Reference

Socket Address.

`#include <unixaddress.h>`

Inheritance diagram for Coral::UnixAddress:



Public Member Functions

- UnixAddress ()
- UnixAddress (const UnixAddress &address)
- UnixAddress (const char *name)
- UnixAddress (sockaddr *address, cardinal length)
- ~UnixAddress ()

- void reset ()
- void init (const UnixAddress &address)
- void init (const char *name)
- UnixAddress & operator= (const UnixAddress &source)
- bool isValid () const
- bool isNull () const
- String getAddressString () const
- cardinal getSystemAddress (sockaddr *buffer, const cardinal length, const cardinal type) const
- bool setSystemAddress (sockaddr *address, const cardinal 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]

Static Private Attributes

- static const cardinal MaxNameLength = 108

6.131.1 Detailed Description

Socket Address.

This class manages an unix socket address.

Author

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

6.131.2 Constructor & Destructor Documentation

6.131.2.1 Coral::UnixAddress::UnixAddress ( )

Constructor for an empty unix address.

6.131.2.2 Coral::UnixAddress::UnixAddress (const UnixAddress & *address*)

Constructor for an unix address from an unix address.

Parameters

address	Unix address.

6.131.2.3 Coral::UnixAddress::UnixAddress (const char * *name*)

Constructor for a unix address given by a string. Examples: "/tmp/test.socket".

Parameters

name	Address string.

6.131.2.4 Coral::UnixAddress::UnixAddress (sockaddr * *address*, cardinal *length*)

Constructor for a unix address from the system's sockaddr structure.

Parameters

address	sockaddr.
length	Length of sockaddr.

6.131.2.5 Coral::UnixAddress::~UnixAddress ( )

Destructor.

6.131.3 Member Function Documentation

6.131.3.1 String Coral::UnixAddress::getAddressString ( ) const [virtual]

Get address string.

Returns

Address string.

Implements Coral::SocketAddress.

6.131.3.2 cardinal Coral::UnixAddress::getSystemAddress (sockaddr * *buffer*, const cardinal *length*, const cardinal *type*) const

getSystemAddress() implementation of SocketAddress

See also

SocketAddress::getSystemAddress

6.131.3.3 void Coral::UnixAddress::init (const UnixAddress & *address*)

Initialize unix address from unix address.

6.131.3.4 void Coral::UnixAddress::init (const char * *name*)

Initialize unix address from socket name.

6.131.3.5 bool Coral::UnixAddress::isNull ( ) const [inline]

Check, if the address is null.

Returns

true, if the address is not null; false otherwise.

6.131.3.6 bool Coral::UnixAddress::isValid ( ) const [virtual]

isValid() implementation of SocketAddress.

See also

SocketAddress::isValid

Implements Coral::SocketAddress.

6.131.3.7 int Coral::UnixAddress::operator!= (const UnixAddress & *address*) const [inline]

Implementation of != operator.

6.131.3.8 int Coral::UnixAddress::operator< (const UnixAddress & *address*) const

Implementation of < operator.

6.131.3.9 int Coral::UnixAddress::operator<= (const UnixAddress & *address*) const [inline]

Implementation of <= operator.

6.131.3.10 UnixAddress& Coral::UnixAddress::operator= (const UnixAddress & *source*) [inline]

Implementation of = operator.

6.131.3.11 int Coral::UnixAddress::operator== (const UnixAddress & *address*) const

Implementation of == operator.

6.131.3.12 int Coral::UnixAddress::operator> (const UnixAddress & *address*) const

Implementation of > operator.

6.131.3.13 int Coral::UnixAddress::operator>= (const UnixAddress & *address*) const [inline]

Implementation of >= operator.

6.131.3.14 void Coral::UnixAddress::reset ( ) [virtual]

Reset unix address.

Implements Coral::SocketAddress.

6.131.3.15 bool Coral::UnixAddress::setSystemAddress (sockaddr * *address*, const cardinal *length*)

setSystemAddress() implementation of SocketAddress.

See also

SocketAddress::setSystemAddress

6.131.4 Member Data Documentation

6.131.4.1 const cardinal Coral::UnixAddress::MaxNameLength = 108 [static, private]

6.131.4.2 char Coral::UnixAddress::Name[MaxNameLength] [private]

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

- unixaddress.h
- unixaddress.cc

6.132 Coral::TraceServer::User Struct Reference

Public Attributes

- RTCPAbstractServer::Client * Client
- RTPSender Sender
- Socket SenderSocket
- InternetFlow Flow
- TraceEncoderRepository Repository
- TDTFMediaReader Reader
- String MediaName
- integer StreamIdentifier
- card16 LastSequenceNumber
- card16 PosChgSeqNumber
- bool ClientPause

6.132.1 Member Data Documentation

6.132.1.1 RTCPAbstractServer::Client* Coral::TraceServer::User::Client

6.132.1.2 bool Coral::TraceServer::User::ClientPause

6.132.1.3 InternetFlow Coral::TraceServer::User::Flow

6.132.1.4 card16 Coral::TraceServer::User::LastSequenceNumber

6.132.1.5 String Coral::TraceServer::User::MediaName

6.132.1.6 card16 Coral::TraceServer::User::PosChgSeqNumber

6.132.1.7 TDTFMediaReader Coral::TraceServer::User::Reader

6.132.1.8 TraceEncoderRepository Coral::TraceServer::User::Repository

6.132.1.9 RTPSender Coral::TraceServer::User::Sender

6.132.1.10 Socket Coral::TraceServer::User::SenderSocket

6.132.1.11 integer Coral::TraceServer::User::StreamIdentifier

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

- traceserver.h

6.133 Coral::UtilizationHeader Struct Reference

Trace Header.

`#include <tdtf.h>`

Static Public Member Functions

- static cardinal getUtilizationSize (const cardinal maxConstants)

Public Attributes

- card16 Type
- card8 MaxConstants
- card8 Constants
- card64 Weight
- card64 Constant [0]

6.133.1 Detailed Description

Trace Header.

This is the header for a trace.

Author

Thomas Dreibholz dreibh@iem.uni-due.de

Version

1.0

6.133.2 Member Function Documentation

6.133.2.1 static cardinal Coral::UtilizationHeader::getUtilizationSize (const cardinal *maxConstants*) [inline, static]

Calculate utilization size.

Parameters

maxConstants	Maximum number of constants.

Returns

Utilization size.

6.133.3 Member Data Documentation

6.133.3.1 card64 Coral::UtilizationHeader::Constant[0]

Array of utilization constants (double converted to binary using translateToBinary()).

See also

translateToBinary
translateToDouble

6.133.3.2 card8 Coral::UtilizationHeader::Constants

Number of utilization constants.

6.133.3.3 card8 Coral::UtilizationHeader::MaxConstants

Maximum number of utilization constants.

6.133.3.4 card16 Coral::UtilizationHeader::Type

Utility function type.

6.133.3.5 card64 Coral::UtilizationHeader::Weight

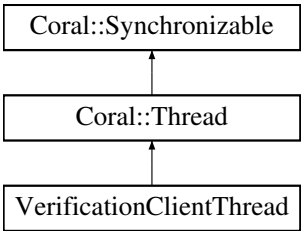
Utilization weight.

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

- tdtf.h

6.134 VerificationClientThread Class Reference

Inheritance diagram for VerificationClientThread:



Public Member Functions

- VerificationClientThread (const cardinal id, const char *server=DefaultServer, const char *media=DefaultMedia, const cardinal count=DefaultMediaCount, const cardinal pause=DefaultPause, const double prSelectEncoding=0.001, const double prSelectQuality=0.02, const double prSelectPosition=0.05, const double prSelectMedia=0.01, const double prStop=0.003, const double prRestart=0.001)
- 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
- TraceClient * Client
- card32 SSRC
- card64 Position
- double Utilization
- double FrameRate
- integer StreamPriority
- integer SessionPriority
- Range< card64 > Bandwidth
- const char * Encoding
- cardinal ID
- const char * Server
- const char * Media
- cardinal MediaCount
- cardinal Pause
- double PrSelectEncoding
- double PrSelectQuality
- double PrSelectPosition
- double PrSelectMedia
- double PrStop
- double PrRestart

6.134.1 Constructor & Destructor Documentation

6.134.1.1 VerificationClientThread::VerificationClientThread (const cardinal *id*, const char * *server* = DefaultServer, const char * *media* = DefaultMedia, const cardinal *count* = DefaultMediaCount, const cardinal *pause* = DefaultPause, const double *prSelectEncoding* = 0.001, const double *prSelectQuality* = 0.02, const double *prSelectPosition* = 0.05, const double *prSelectMedia* = 0.01, const double *prStop* = 0.003, const double *prRestart* = 0.001)

6.134.2 Member Function Documentation

6.134.2.1 char * VerificationClientThread::getStatusString (char * *str*, const cardinal *maxLength*)

6.134.2.2 void VerificationClientThread::restart ( ) [private]

6.134.2.3 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 Coral::Thread.

6.134.2.4 void VerificationClientThread::selectEncoding ( ) [private]

6.134.2.5 void VerificationClientThread::selectMedia ( ) [private]

6.134.2.6 void VerificationClientThread::selectPosition ( ) [private]

6.134.2.7 void VerificationClientThread::selectQuality ( ) [private]

6.134.2.8 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

Reimplemented from Coral::Thread.

6.134.2.9 void VerificationClientThread::test ( ) [private]

6.134.2.10 void VerificationClientThread::writeLog (const char * *str*) [private]

6.134.3 Member Data Documentation

6.134.3.1 Range<card64> VerificationClientThread::Bandwidth [private]

6.134.3.2 TraceClient* VerificationClientThread::Client [private]

6.134.3.3 const char* VerificationClientThread::Encoding [private]

6.134.3.4 double VerificationClientThread::FrameRate [private]

6.134.3.5 cardinal VerificationClientThread::ID [private]

6.134.3.6 const char* VerificationClientThread::Media [private]

6.134.3.7 cardinal VerificationClientThread::MediaCount [private]

6.134.3.8 cardinal VerificationClientThread::Pause [private]

6.134.3.9 card64 VerificationClientThread::Position [private]

6.134.3.10 double VerificationClientThread::PrRestart [private]

6.134.3.11 double VerificationClientThread::PrSelectEncoding [private]

6.134.3.12 double VerificationClientThread::PrSelectMedia [private]

6.134.3.13 double VerificationClientThread::PrSelectPosition [private]

6.134.3.14 double VerificationClientThread::PrSelectQuality [private]

6.134.3.15 double VerificationClientThread::PrStop [private]

6.134.3.16 Randomizer VerificationClientThread::Random [private]

6.134.3.17 const char* VerificationClientThread::Server [private]

6.134.3.18 integer VerificationClientThread::SessionPriority [private]

6.134.3.19 card32 VerificationClientThread::SSRC [private]

6.134.3.20 integer VerificationClientThread::StreamPriority [private]

6.134.3.21 double VerificationClientThread::Utilization [private]

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

- vtclient.cc

Chapter 7

File Documentation

7.1 abstractlayerdescription.cc File Reference

```
#include "system.h"   #include "abstractlayerdescription.-
h"
```

Namespaces

- namespace Coral

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 "system.h" #include "framesizescalabilityinterface.-
h" #include "internetflow.h" #include "abstractlayerdescription.-
icc"
```

Classes

- class Coral::AbstractLayerDescription

  Abstract Layer Description.

Namespaces

- namespace Coral

7.3 abstractqosdescription.cc File Reference

```
#include "system.h" #include "abstractqosdescription.h"
```

Namespaces

- namespace Coral

Functions

- ostream & Coral::operator<< (ostream &os, const AbstractQoSDescription &aqd)

7.4 abstractqosdescription.h File Reference

```
#include "system.h" #include "frameratescalabilityinterface.-
h" #include "abstractlayerdescription.h" #include "rtppacket.-
h" #include "resourceutilizationpoint.h" #include "abstractqosdescription.-
icc"
```

Classes

- class Coral::AbstractQoSDescription

  Abstract QoS Description.

Namespaces

- namespace Coral

Functions

- ostream & Coral::operator<< (ostream &os, const AbstractQoSDescription &aqd)

7.5 averageanalyzer.cc File Reference

```
#include "system.h" #include "tools.h" #include "strings.-
h" #include <fstream.h> #include <time.h>
```

Functions

- int main (int argc, char *argv[])

7.5.1 Function Documentation

7.5.1.1 int main (int *argc*, char * *argv[]*)

7.6 bandwidthinfo.cc File Reference

```
#include "system.h" #include "bandwidthinfo.h"
```

Namespaces

- namespace Coral

Functions

- ostream & Coral::operator<< (ostream &os, const BandwidthInfo &bi)

7.7 bandwidthinfo.h File Reference

```
#include "system.h" #include "bandwidthinfo.icc"
```

Classes

- struct Coral::BandwidthInfo

  Bandwidth Info.

Namespaces

- namespace Coral

Functions

- ostream & Coral::operator<< (ostream &os, const BandwidthInfo &bi)

7.8 bandwidthmanager.cc File Reference

```
#include "system.h" #include "bandwidthmanager.h" #include
"streamdescription.h"
```

Namespaces

- namespace Coral

Functions

- ostream & Coral::operator<< (ostream &os, const ResourceUtilizationSimplePoint &srup)
- ostream & Coral::operator<< (ostream &os, const ResourceUtilizationMultiPoint &srup)

7.9 bandwidthmanager.h File Reference

```
#include "system.h"  #include "abstractqosdescription.h" ×
#include "timedthread.h" #include "servicelevelagreement.-
h" #include "streamdescription.h" #include "sessiondescription.-
h" #include "roundtriptimepinger.h" #include "rtcppacket.-
h" #include <multimap.h> #include <algo.h> #include "bandwidthmanager.-
icc"
```

Classes

- struct Coral::ResourceUtilizationSimplePoint

  Resource Utilization Simple Point.
- struct Coral::ResourceUtilizationMultiPoint

  Resource Utilization Simple Point.
- class Coral::BandwidthManager

  Bandwidth Manager.

Namespaces

- namespace Coral

Functions

- ostream & Coral::operator<< (ostream &os, const ResourceUtilizationSimplePoint &srup)

- ostream & Coral::operator<< (ostream &os, const ResourceUtilizationMultiPoint &srup)

7.10 breakdetector.cc File Reference

```
#include "system.h"  #include "breakdetector.h"  #include
<signal.h>
```

Namespaces

- namespace Coral

Functions

- void Coral::breakDetector (int signum)
- void Coral::installBreakDetector ()
- void Coral::uninstallBreakDetector ()
- bool Coral::breakDetected ()

Variables

- bool Coral::DetectedBreak = false
- bool Coral::PrintedBreak = false

7.11 breakdetector.h File Reference

```
#include "system.h"
```

Namespaces

- namespace Coral

Functions

- void Coral::installBreakDetector ()
- void Coral::uninstallBreakDetector ()
- bool Coral::breakDetected ()

7.12 cbrframesizescalability.cc File Reference

```
#include "system.h" #include "cbrframesizescalability.h"
```

Namespaces

- namespace Coral

7.13 cbrframesizescalability.h File Reference

`#include "system.h" #include "framesizescalabilityinterface.h" #include "genericframesizescalability.h"`

Classes

- class Coral::ConstantBitrateFrameSizeScalability

  Constant Bitrate Frame Size Scalability.

Namespaces

- namespace Coral

7.14 decoderinterface.cc File Reference

`#include "system.h" #include "decoderinterface.h"`

Namespaces

- namespace Coral

7.15 decoderinterface.h File Reference

`#include "system.h" #include "sourcestateinfo.h" #include "mediainfo.h"`

Classes

- struct Coral::DecoderPacket

  DecoderPacket.
- class Coral::DecoderInterface

  Decoder Interface.

Namespaces

- namespace Coral

7.16 decoderrepositoryinterface.h File Reference

```
#include "system.h"  #include "synchronizable.h"  #include
"decoderinterface.h"
```

Classes

- class Coral::DecoderRepositoryInterface

  Decoder Repository.

Namespaces

- namespace Coral

7.17 delayanalyzer.cc File Reference

```
#include "system.h" #include "tools.h" #include "strings.-
h" #include <fstream.h> #include <time.h>
```

Classes

- struct MediaList

Functions

- int main (int argc, char ∗argv[])

Variables

- const cardinal Entries = 100
- MediaList MList [Entries]

7.17.1 Function Documentation

7.17.1.1 int main (int *argc,* char ∗ *argv[]*)

7.17.2 Variable Documentation

7.17.2.1 const cardinal Entries = 100

7.17.2.2 MediaList MList[Entries]

7.18 differenceanalyzer.cc File Reference

```
#include "system.h" #include "tools.h" #include "strings.-
h" #include <fstream.h> #include <time.h>
```

Classes

- struct MediaList

Functions

- int main (int argc, char *argv[])

Variables

- const cardinal Entries = 100
- MediaList MList [Entries]

7.18.1 Function Documentation

7.18.1.1 int main (int *argc,* char * *argv[]*)

7.18.2 Variable Documentation

7.18.2.1 const cardinal Entries = 100

7.18.2.2 MediaList MList[Entries]

7.19 encoderinterface.cc File Reference

```
#include "system.h" #include "encoderinterface.h"
```

Namespaces

- namespace Coral

7.20 encoderinterface.h File Reference

```
#include "system.h" #include "range.h" #include "abstractqosdescription.-
h"
```

Classes

- struct Coral::EncoderPacket

  EncoderPacket.
- class Coral::EncoderInterface

  Encoder Interface.

Namespaces

- namespace Coral

7.21 encoderrepositoryinterface.h File Reference

```
#include "system.h"  #include "synchronizable.h"  #include
"encoderinterface.h"
```

Classes

- class Coral::EncoderRepositoryInterface

  Encoder Repository Interface.

Namespaces

- namespace Coral

7.22 frameratescalabilityinterface.h File Reference

```
#include "system.h"
```

Classes

- class Coral::FrameRateScalabilityInterface

  Frame Rate Scalability Interface.

Namespaces

- namespace Coral

7.23 framesizescalabilityinterface.h File Reference

```
#include "system.h"
```

Classes

- class Coral::FrameSizeScalabilityInterface

  Frame Rate Scalability Interface.

Namespaces

- namespace Coral

7.24 genericframesizescalability.cc File Reference

```
#include "system.h" #include "genericframesizescalability.-
h"
```

Namespaces

- namespace Coral

7.25 genericframesizescalability.h File Reference

```
#include "system.h" #include "framesizescalabilityinterface.-
h"
```

Classes

- class Coral::GenericFrameSizeScalability

  Generic Frame Size Scalability.

Namespaces

- namespace Coral

7.26 globaltraceconfiguration.cc File Reference

```
#include "system.h"  #include "globaltraceconfiguration.-
h"
```

Namespaces

- namespace Coral

Variables

- TraceConfiguration Coral::TraceConfig

7.27 globaltraceconfiguration.h File Reference

```
#include "system.h" #include "traceconfiguration.h"
```

Namespaces

- namespace Coral

7.28 gnuplot.cc File Reference

```
#include "system.h" #include "gnuplot.h" #include "tools.-
h" #include <time.h> #include <fstream.h>
```

Namespaces

- namespace Coral

Defines

- #define GNUPLOT_H

7.28.1 Define Documentation

7.28.1.1 #define GNUPLOT_H

7.29 gnuplot.h File Reference

`#include "system.h" #include "strings.h" #include <fstream.-h>`

Classes

- class Coral::GNUPlotData

  GNUPlot Data.
- class Coral::GNUPlotScript

  GNUPlot Data.

Namespaces

- namespace Coral

7.30 h263qosdescription.cc File Reference

`#include "system.h" #include "h263qosdescription.h"`

Namespaces

- namespace Coral

Defines

- #define VERIFY_CALCULATION

7.30.1 Define Documentation

7.30.1.1 #define VERIFY_CALCULATION

7.31 h263qosdescription.h File Reference

`#include "system.h" #include "traceqosdescription.h"`

Classes

- class Coral::H263QoSDescription

  H263 QoS Description.

Namespaces

- namespace Coral

7.32 h263totrace.cc File Reference

```
#include "system.h" #include "tools.h" #include <sys/mman.-
h> #include <fcntl.h> #include <limits.h>
```

Functions

- cardinal getBit (const char *data, const cardinal bit)
- cardinal getBlock (const char *data, const cardinal bit, const cardinal length)
- void printBinary (const char *byte)
- void getTrace (const char *data, const cardinal length)
- int main (int argc, char **argv)

Variables

- const char * Format [8]

7.32.1 Function Documentation

7.32.1.1 cardinal getBit (const char * *data*, const cardinal *bit*) [inline]

7.32.1.2 cardinal getBlock (const char * *data*, const cardinal *bit*, const cardinal *length*) [inline]

7.32.1.3 void getTrace (const char * *data*, const cardinal *length*)

7.32.1.4 int main (int *argc*, char ** *argv*)

7.32.1.5 void printBinary (const char * *byte*)

7.32.2 Variable Documentation

7.32.2.1 const char* Format[8]

Initial value:

```
{
   "", "sub-QCIF", "QCIF", "CIF", "4CIF", "16CIF", "?", "extended PTYPE"
}
```

7.33 h263tracearray.cc File Reference

```
#include "system.h"  #include "h263tracearray.h"  #include
"randomizer.h" #include <fstream.h>
```

Namespaces

- namespace Coral

7.34 h263tracearray.h File Reference

```
#include "system.h" #include "tracearray.h"
```

Classes

- class Coral::H263TraceArray

  H263 Trace Array.

Namespaces

- namespace Coral

7.35 h263writerqosdescription.cc File Reference

```
#include "system.h"  #include "h263writerqosdescription.-
h" #include "utilityfunction.h"
```

Namespaces

- namespace Coral

7.36 h263writerqosdescription.h File Reference

```
#include "system.h" #include "h263qosdescription.h" #include
"traceconfiguration.h"
```

Classes

- class Coral::H263WriterQoSDescription

  H263 Writer QoS Description.

Namespaces

- namespace Coral

7.37 in6.h File Reference

```
#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 IPPROTO_HOPOPTS 0 /∗ IPv6 hop-by-hop options ∗/
- #define IPPROTO_ROUTING 43 /∗ IPv6 routing header ∗/

- #define IPPROTO_FRAGMENT 44 /∗ IPv6 fragmentation header ∗/
- #define IPPROTO_ICMPV6 58 /∗ ICMPv6 ∗/
- #define IPPROTO_NONE 59 /∗ IPv6 no next header ∗/
- #define IPPROTO_DSTOPTS 60 /∗ IPv6 destination options ∗/
- #define IPV6_TLV_PAD0 0
- #define IPV6_TLV_PADN 1
- #define IPV6_TLV_ROUTERALERT 20
- #define IPV6_TLV_JUMBO 194
- #define IPV6_ADDRFORM 1
- #define IPV6_PKTINFO 2
- #define IPV6_HOPOPTS 3
- #define IPV6_DSTOPTS 4
- #define IPV6_RTHDR 5
- #define IPV6_PKTOPTIONS 6
- #define IPV6_CHECKSUM 7
- #define IPV6_HOPLIMIT 8
- #define IPV6_NEXTHOP 9
- #define IPV6_AUTHHDR 10
- #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_ADD_MEMBERSHIP 20
- #define IPV6_DROP_MEMBERSHIP 21
- #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
- #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 IPPROTO_HOPOPTS 0 /∗ IPv6 hop-by-hop options ∗/
- #define IPPROTO_ROUTING 43 /∗ IPv6 routing header ∗/
- #define IPPROTO_FRAGMENT 44 /∗ IPv6 fragmentation header ∗/
- #define IPPROTO_ICMPV6 58 /∗ ICMPv6 ∗/
- #define IPPROTO_NONE 59 /∗ IPv6 no next header ∗/
- #define IPPROTO_DSTOPTS 60 /∗ IPv6 destination options ∗/
- #define IPV6_TLV_PAD0 0
- #define IPV6_TLV_PADN 1
- #define IPV6_TLV_ROUTERALERT 20
- #define IPV6_TLV_JUMBO 194
- #define IPV6_ADDRFORM 1
- #define IPV6_PKTINFO 2
- #define IPV6_HOPOPTS 3
- #define IPV6_DSTOPTS 4
- #define IPV6_RTHDR 5
- #define IPV6_PKTOPTIONS 6
- #define IPV6_CHECKSUM 7
- #define IPV6_HOPLIMIT 8
- #define IPV6_NEXTHOP 9
- #define IPV6_AUTHHDR 10
- #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_ADD_MEMBERSHIP 20
- #define IPV6_DROP_MEMBERSHIP 21
- #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

7.37.1 Define Documentation

7.37.1.1 #define IPPROTO_DSTOPTS 60 /* IPv6 destination options */

7.37.1.2 #define IPPROTO_DSTOPTS 60 /* IPv6 destination options */

7.37.1.3 #define IPPROTO_FRAGMENT 44 /* IPv6 fragmentation header */

7.37.1.4 #define IPPROTO_FRAGMENT 44 /* IPv6 fragmentation header */

7.37.1.5 #define IPPROTO_HOPOPTS 0 /* IPv6 hop-by-hop options */

7.37.1.6 #define IPPROTO_HOPOPTS 0 /* IPv6 hop-by-hop options */

7.37.1.7 #define IPPROTO_ICMPV6 58 /* ICMPv6 */

7.37.1.8 #define IPPROTO_ICMPV6 58 /* ICMPv6 */

7.37.1.9 #define IPPROTO_NONE 59 /* IPv6 no next header */

7.37.1.10 #define IPPROTO_NONE 59 /* IPv6 no next header */

7.37.1.11 #define IPPROTO_ROUTING 43 /* IPv6 routing header */

7.37.1.12 #define IPPROTO_ROUTING 43 /* IPv6 routing header */

7.37.1.13 #define IPV6_ADD_MEMBERSHIP 20

7.37.1.14 #define IPV6_ADD_MEMBERSHIP 20

7.37.1.15 #define IPV6_ADDRFORM 1

7.37.1.16 #define IPV6_ADDRFORM 1

7.37.1.17 #define IPV6_AUTHHDR 10

7.37.1.18 #define IPV6_AUTHHDR 10

7.37.1.19 #define IPV6_CHECKSUM 7

7.37.1.20 #define IPV6_CHECKSUM 7

7.37.1.21 #define IPV6_DROP_MEMBERSHIP 21

7.37.1.22 #define IPV6_DROP_MEMBERSHIP 21

7.37.1.23 #define IPV6_DSTOPTS 4

7.37.1.24 #define IPV6_DSTOPTS 4

7.37.1.25 #define IPV6_FL_A_GET 0

7.37.1.26 #define IPV6_FL_A_GET 0

7.37.1.27 #define IPV6_FL_A_PUT 1

7.37.1.28 #define IPV6_FL_A_PUT 1

7.37.1.29 #define IPV6_FL_A_RENEW 2

7.37.1.30 #define IPV6_FL_A_RENEW 2

7.37.1.31 #define IPV6_FL_F_CREATE 1

7.37.1.32 #define IPV6_FL_F_CREATE 1

7.37.1.33 #define IPV6_FL_F_EXCL 2

7.37.1.34 #define IPV6_FL_F_EXCL 2

7.37.1.35 #define IPV6_FL_S_ANY 255

7.37.1.36 #define IPV6_FL_S_ANY 255

7.37.1.37 #define IPV6_FL_S_EXCL 1

7.37.1.38 #define IPV6_FL_S_EXCL 1

7.37.1.39 #define IPV6_FL_S_NONE 0

7.37.1.40 #define IPV6_FL_S_NONE 0

7.37.1.41 #define IPV6_FL_S_PROCESS 2

7.37.1.42 #define IPV6_FL_S_PROCESS 2

7.37.1.43 #define IPV6_FL_S_USER 3

7.37.1.44 #define IPV6_FL_S_USER 3

7.37.1.45 #define IPV6_FLOWINFO 11

7.37.1.46 #define IPV6_FLOWINFO 11

7.37.1.47 #define IPV6_FLOWINFO_FLOWLABEL 0x000fffff

7.37.1.48 #define IPV6_FLOWINFO_FLOWLABEL 0x000fffff

7.37.1.49 #define IPV6_FLOWINFO_PRIORITY 0x0ff00000

7.37.1.50 #define IPV6_FLOWINFO_PRIORITY 0x0ff00000

7.37.1.51 #define IPV6_FLOWINFO_SEND 33

7.37.1.52 #define IPV6_FLOWINFO_SEND 33

7.37.1.53 #define IPV6_FLOWLABEL_MGR 32

7.37.1.54 #define IPV6_FLOWLABEL_MGR 32

7.37.1.55 #define IPV6_HOPLIMIT 8

7.37.1.56 #define IPV6_HOPLIMIT 8

7.37.1.57 #define IPV6_HOPOPTS 3

7.37.1.58 #define IPV6_HOPOPTS 3

7.37.1.59 #define IPV6_MTU 24

7.37.1.60 #define IPV6_MTU 24

7.37.1.61 #define IPV6_MTU_DISCOVER 23

7.37.1.62 #define IPV6_MTU_DISCOVER 23

7.37.1.63 #define IPV6_MULTICAST_HOPS 18

7.37.1.64 #define IPV6_MULTICAST_HOPS 18

7.37.1.65 #define IPV6_MULTICAST_IF 17

7.37.1.66 #define IPV6_MULTICAST_IF 17

7.37.1.67 #define IPV6_MULTICAST_LOOP 19

7.37.1.68 #define IPV6_MULTICAST_LOOP 19

7.37.1.69 #define IPV6_NEXTHOP 9

7.37.1.70 #define IPV6_NEXTHOP 9

7.37.1.71 #define IPV6_PKTINFO 2

7.37.1.72 #define IPV6_PKTINFO 2

7.37.1.73 #define IPV6_PKTOPTIONS 6

7.37.1.74 #define IPV6_PKTOPTIONS 6

7.37.1.75 #define IPV6_PMTUDISC_DO 2

7.37.1.76 #define IPV6_PMTUDISC_DO 2

7.37.1.77 #define IPV6_PMTUDISC_DONT 0

7.37.1.78 #define IPV6_PMTUDISC_DONT 0

7.37.1.79 #define IPV6_PMTUDISC_WANT 1

7.37.1.80 #define IPV6_PMTUDISC_WANT 1

7.37.1.81 #define IPV6_PRIORITY_10 0x0a00

7.37.1.82 #define IPV6_PRIORITY_10 0x0a00

7.37.1.83 #define IPV6_PRIORITY_11 0x0b00

7.37.1.84 #define IPV6_PRIORITY_11 0x0b00

7.37.1.85 #define IPV6_PRIORITY_12 0x0c00

7.37.1.86 #define IPV6_PRIORITY_12 0x0c00

7.37.1.87 #define IPV6_PRIORITY_13 0x0d00

7.37.1.88 #define IPV6_PRIORITY_13 0x0d00

7.37.1.89 #define IPV6_PRIORITY_14 0x0e00

7.37.1.90 #define IPV6_PRIORITY_14 0x0e00

7.37.1.91 #define IPV6_PRIORITY_15 0x0f00

7.37.1.92 #define IPV6_PRIORITY_15 0x0f00

7.37.1.93 #define IPV6_PRIORITY_8 0x0800

7.37.1.94 #define IPV6_PRIORITY_8 0x0800

7.37.1.95 #define IPV6_PRIORITY_9 0x0900

7.37.1.96 #define IPV6_PRIORITY_9 0x0900

7.37.1.97 #define IPV6_PRIORITY_BULK 0x0400

7.37.1.98 #define IPV6_PRIORITY_BULK 0x0400

7.37.1.99 #define IPV6_PRIORITY_CONTROL 0x0700

7.37.1.100 #define IPV6_PRIORITY_CONTROL 0x0700

7.37.1.101 #define IPV6_PRIORITY_FILLER 0x0100

7.37.1.102 #define IPV6_PRIORITY_FILLER 0x0100

7.37.1.103 #define IPV6_PRIORITY_INTERACTIVE 0x0600

7.37.1.104 #define IPV6_PRIORITY_INTERACTIVE 0x0600

7.37.1.105 #define IPV6_PRIORITY_RESERVED1 0x0300

7.37.1.106 #define IPV6_PRIORITY_RESERVED1 0x0300

7.37.1.107 #define IPV6_PRIORITY_RESERVED2 0x0500

7.37.1.108 #define IPV6_PRIORITY_RESERVED2 0x0500

7.37.1.109 #define IPV6_PRIORITY_UNATTENDED 0x0200

7.37.1.110 #define IPV6_PRIORITY_UNATTENDED 0x0200

7.37.1.111 #define IPV6_PRIORITY_UNCHARACTERIZED 0x0000

7.37.1.112 #define IPV6_PRIORITY_UNCHARACTERIZED 0x0000

7.37.1.113 #define IPV6_RECVERR 25

7.37.1.114 #define IPV6_RECVERR 25

7.37.1.115 #define IPV6_ROUTER_ALERT 22

7.37.1.116 #define IPV6_ROUTER_ALERT 22

7.37.1.117 #define IPV6_RTHDR 5

7.37.1.118 #define IPV6_RTHDR 5

7.37.1.119 #define IPV6_TLV_JUMBO 194

7.37.1.120 #define IPV6_TLV_JUMBO 194

7.37.1.121 #define IPV6_TLV_PAD0 0

7.37.1.122 #define IPV6_TLV_PAD0 0

7.37.1.123 #define IPV6_TLV_PADN 1

7.37.1.124 #define IPV6_TLV_PADN 1

7.37.1.125 #define IPV6_TLV_ROUTERALERT 20

7.37.1.126 #define IPV6_TLV_ROUTERALERT 20

7.37.1.127 #define IPV6_UNICAST_HOPS 16

7.37.1.128 #define IPV6_UNICAST_HOPS 16

7.37.1.129 #define SCM_SRCRT IPV6_RXSRCRT

7.37.1.130 #define SCM_SRCRT IPV6_RXSRCRT

7.38 internetaddress.cc File Reference

```
#include "system.h" #include "strings.h" #include "socket.-
h" #include <netdb.h> #include <netinet/in.h>
```

Namespaces

- namespace Coral

7.39 internetaddress.h File Reference

```
#include "system.h" #include "strings.h" #include "socketaddress.-
h" #include "portableaddress.h" #include <netinet/in.h> ×
```

```
#include <resolv.h> #include "internetaddress.icc"
```

Classes

- class Coral::InternetAddress

  Socket Address.

Namespaces

- namespace Coral

7.40 internetflow.cc File Reference

```
#include "system.h" #include "internetaddress.h" #include
"internetflow.h"
```

Namespaces

- namespace Coral

7.41 internetflow.h File Reference

```
#include "system.h" #include "internetaddress.h" #include
"internetflow.icc"
```

Classes

- class Coral::InternetFlow

  Internet Flow.
- struct Coral::in6_flowlabel_req

Namespaces

- namespace Coral
- namespace Coral::Coral

7.42 loganalyzer.cc File Reference

```
#include "system.h" #include "tools.h" #include "strings.-
h" #include "trafficclassvalues.h" #include "servicelevelagreement.-
h" #include <fstream.h> #include <time.h> #include <set.-
h> #include <algo.h>
```

Classes

- struct StreamEntry
- struct ClassEntry

Functions

- StreamEntry * findStream (const card64 id)
- int main (int argc, char *argv[])

Variables

- const cardinal Entries = 1000
- set< StreamEntry > StreamList
- ClassEntry ClassList [TrafficClassValues::MaxValues]

7.42.1 Function Documentation

7.42.1.1 StreamEntry* findStream (const card64 *id*)

7.42.1.2 int main (int *argc,* char * *argv[]*)

7.42.2 Variable Documentation

7.42.2.1 ClassEntry ClassList[TrafficClassValues::MaxValues]

7.42.2.2 const cardinal Entries = 1000

7.42.2.3 set<StreamEntry> StreamList

7.43 managedstreaminterface.h File Reference

```
#include "system.h" #include "abstractqosdescription.h"
```

Classes

- class Coral::ManagedStreamInterface

  Managed Stream Interface.

Namespaces

- namespace Coral

7.44 mediainfo.cc File Reference

```
#include "system.h" #include "mediainfo.h" #include "tools.-
h"
```

Namespaces

- namespace Coral

Functions

- ostream & Coral::operator<< (ostream &os, const MediaInfo &mi)

7.45 mediainfo.h File Reference

```
#include "tools.h"
```

Classes

- class Coral::MediaInfo

  Media Info.

Namespaces

- namespace Coral

Enumerations

- enum Coral::MediaError { Coral::ME_NoError = 0, Coral::ME_NoMedia = 1, -Coral::ME_EOF = 2, Coral::ME_UnrecoverableError = 20, Coral::ME_BadMedia = ME_UnrecoverableError + 0, Coral::ME_ReadError = ME_UnrecoverableError + 1, Coral::ME_OutOfMemory = ME_UnrecoverableError + 2 }

Functions

- ostream & Coral::operator<< (ostream &os, const MediaInfo &mi)

Variables

- const card64 Coral::PositionStepsPerSecond = (card64)1000000000

7.46 mp3qosdescription.cc File Reference

```
#include "system.h" #include "mp3qosdescription.h"
```

Namespaces

- namespace Coral

7.47 mp3qosdescription.h File Reference

```
#include "system.h" #include "traceqosdescription.h"
```

Classes

- class Coral::MP3QoSDescription

  MP3 QoS Description.

Namespaces

- namespace Coral

7.48 mp3totrace.cc File Reference

```
#include "system.h" #include <fstream.h>
```

Enumerations

- enum _mpegversion { mpeg1, mpeg2 }
- enum _mode { fullstereo, jocardinal, dual, single }
- enum _frequency { frequency44100, frequency48000, frequency32000 }

Functions

- cardinal readByte (ifstream ∗loader)
- bool skipBytes (ifstream ∗loader, const cardinal bytes)
- bool isValidHeader (const cardinal mpeg, const cardinal mylayer, const cardinal brindex, const cardinal sfreq)
- bool getNextFrame (ifstream ∗loader)
- int main (int argc, char ∗∗argv)

Variables

- const cardinal frequencies [2][3]
- const cardinal bitrate [2][3][15]
- cardinal layer3slots
- cardinal layer
- cardinal protection
- cardinal bitrateindex
- cardinal padding
- cardinal extendedmode
- cardinal framesize
- cardinal frames = 0
- enum _mpegversion version
- enum _mode mode
- enum _frequency frequency

7.48.1 Enumeration Type Documentation

7.48.1.1 enum _frequency

Enumerator:

frequency44100

frequency48000

frequency32000

7.48.1.2 enum _mode

Enumerator:

fullstereo

jocardinal

dual

single

7.48.1.3 enum _mpegversion

Enumerator:

mpeg1

mpeg2

7.48.2 Function Documentation

7.48.2.1 bool getNextFrame (ifstream * *loader*)

7.48.2.2 bool isValidHeader (const cardinal *mpeg,* const cardinal *mylayer,* const cardinal *brindex,* const cardinal *sfreq*)

7.48.2.3 int main (int *argc,* char ** *argv*)

7.48.2.4 cardinal readByte (ifstream * *loader*)

7.48.2.5 bool skipBytes (ifstream * *loader,* const cardinal *bytes*)

7.48.3 Variable Documentation

7.48.3.1 const cardinal bitrate[2][3][15]

Initial value:

```
{

  {{0,32,64,96,128,160,192,224,256,288,320,352,384,416,448},
   {0,32,48,56,64,80,96,112,128,160,192,224,256,320,384},
   {0,32,40,48,56,64,80,96,112,128,160,192,224,256,320}},


  {{0,32,48,56,64,80,96,112,128,144,160,176,192,224,256},
   {0,8,16,24,32,40,48,56,64,80,96,112,128,144,160},
   {0,8,16,24,32,40,48,56,64,80,96,112,128,144,160}}
}
```

7.48.3.2 cardinal bitrateindex

7.48.3.3 cardinal extendedmode

7.48.3.4 cardinal frames = 0

7.48.3.5 cardinal framesize

7.48.3.6 const cardinal frequencies[2][3]

Initial value:

```
{
  {44100,48000,32000},
  {22050,24000,16000}
}
```

7.48.3.7 enum _frequency frequency

7.48.3.8 cardinal layer

7.48.3.9 cardinal layer3slots

7.48.3.10 enum _mode mode

7.48.3.11 cardinal padding

7.48.3.12 cardinal protection

7.48.3.13 enum _mpegversion version

7.49 mp3tracearray.cc File Reference

```
#include "system.h"  #include "mp3tracearray.h"  #include
"randomizer.h" #include <fstream.h>
```

Namespaces

- namespace Coral

7.50 mp3tracearray.h File Reference

```
#include "system.h" #include "tracearray.h"
```

Classes

- class Coral::MP3TraceArray

  MP3 Trace Array.

Namespaces

- namespace Coral

7.51 mp3writerqosdescription.cc File Reference

```
#include "system.h"   #include "mp3writerqosdescription.h"
#include "utilityfunction.h"
```

Namespaces

- namespace Coral

7.52 mp3writerqosdescription.h File Reference

```
#include "system.h" #include "mp3qosdescription.h" #include
"traceconfiguration.h"
```

Classes

- class Coral::MP3WriterQoSDescription

  MP3 Writer QoS Description.

Namespaces

- namespace Coral

7.53 mpegqosdescription.cc File Reference

```
#include "system.h" #include "mpegqosdescription.h"
```

Namespaces

- namespace Coral

Defines

- #define VERIFY_CALCULATION

7.53.1 Define Documentation

7.53.1.1 #define VERIFY_CALCULATION

7.54 mpegqosdescription.h File Reference

```
#include "system.h" #include "traceqosdescription.h"
```

Classes

- class Coral::MPEGQoSDescription

  MPEG QoS Description.

Namespaces

- namespace Coral

7.55 mpegtracearray.cc File Reference

```
#include "system.h"  #include "mpegtracearray.h"  #include
"randomizer.h" #include <fstream.h>
```

Namespaces

- namespace Coral

7.56 mpegtracearray.h File Reference

```
#include "system.h" #include "tracearray.h"
```

Classes

- class Coral::MPEGTraceArray

  MPEG Trace Array.

Namespaces

- namespace Coral

7.57 mpegwriterqosdescription.cc File Reference

```
#include "system.h"  #include "mpegwriterqosdescription.-
h" #include "utilityfunction.h"
```

Namespaces

- namespace Coral

7.58 mpegwriterqosdescription.h File Reference

```
#include "system.h" #include "mpegqosdescription.h" #include
"traceconfiguration.h"
```

Classes

- class Coral::MPEGWriterQoSDescription

  MPEG Writer QoS Description.

Namespaces

- namespace Coral

7.59 pingerhost.h File Reference

```
#include "system.h" #include "socket.h" #include "strings.-
h"  #include "internetaddress.h"  #include "timedthread.h"
#include "pingerhost.icc"
```

Classes

- struct Coral::PingerHost

  PingerHost.

Namespaces

- namespace Coral

Functions

- int Coral::operator== (const PingerHost &ph1, const PingerHost &ph2)
- int Coral::operator< (const PingerHost &ph1, const PingerHost &ph2)
- int Coral::operator> (const PingerHost &ph1, const PingerHost &ph2)

7.60 portableaddress.h File Reference

```
#include "system.h" #include "portableaddress.icc"
```

Classes

- class Coral::PortableAddress

  Portable Internet Address.

Namespaces

- namespace Coral

7.61 randomizer.cc File Reference

```
#include "system.h" #include "randomizer.h" #include "tools.-
h"
```

Namespaces

- namespace Coral

7.62 randomizer.h File Reference

```
#include "system.h" #include "randomizer.icc"
```

Classes

- class Coral::Randomizer

  Randomizer.

Namespaces

- namespace Coral

7.63 range.cc File Reference

```
#include "system.h" #include "range.h"
```

Functions

- template<class T >
  ostream & operator<< (ostream &os, const Range< T > &range)

7.63.1 Function Documentation

7.63.1.1 template<class T > ostream& operator<< (ostream & *os,* const Range< T > & *range*)

<< operator.

7.64 range.h File Reference

```
#include "system.h" #include "range.icc" #include "range.-
cc"
```

Classes

- class Range< T >
  Range.

Functions

- template<class T >
  ostream & operator<< (ostream &os, const Range< T > &range)

7.64.1 Function Documentation

7.64.1.1 template<class T > ostream& operator<< (ostream & *os,* const Range< T > & *range*)

<< operator.

7.65 resourceutilizationpoint.cc File Reference

```
#include "system.h" #include "rtppacket.h" #include "resourceutilizationpoint.-
h"
```

Namespaces

- namespace Coral

Functions

- ostream & Coral::operator<< (ostream &os, const ResourceUtilizationPoint &rup)

7.66 resourceutilizationpoint.h File Reference

```
#include "system.h" #include "rtppacket.h" #include "bandwidthinfo.-
h" #include "trafficclassvalues.h" #include "resourceutilizationpoint.-
icc"
```

Classes

- struct Coral::LayerClassMappingPossibility

  Layer Class Mapping Possibility.
- struct Coral::LayerClassMapping

  Layer Class Mapping.
- class Coral::ResourceUtilizationPoint

  Resource Utilization Point.

Namespaces

- namespace Coral

Functions

- ostream & Coral::operator<< (ostream &os, const ResourceUtilizationPoint &rup)

7.67 roundtriptimepinger.cc File Reference

```
#include "system.h"     #include "roundtriptimepinger.h" ×
#include "socket.h" #include "internetaddress.h" #include
"breakdetector.h" #include "timedthread.h" #include "rtppacket.-
h" #include "strings.h" #include "trafficclassvalues.h"
```

Classes

- struct Coral::icmp_filter

Namespaces

- namespace Coral

Defines

- #define ICMP_FILTER 1

Functions

- ostream & Coral::operator<< (ostream &os, RoundTripTimePinger &pinger)

7.67.1 Define Documentation

7.67.1.1 #define ICMP_FILTER 1

7.68 roundtriptimepinger.h File Reference

```
#include "system.h" #include "socket.h" #include "internetaddress.-
h" #include "timedthread.h" #include "rtppacket.h" #include
"pingerhost.h" #include "randomizer.h" #include <multiset.-
h>  #include <algo.h>  #include <netinet/ip.h>  #include
<netinet/ip_icmp.h>  #include <netinet/icmp6.h>  #include
<sys/time.h> #include <fstream.h> #include "roundtriptimepinger.-
icc"
```

Classes

- class Coral::RoundTripTimePinger

  Round Trip Time Pinger.
- struct Coral::RoundTripTimePinger::Ping4Packet
- struct Coral::RoundTripTimePinger::Ping6Packet

Namespaces

- namespace Coral

7.69 rtcpabstractserver.cc File Reference

```
#include "system.h" #include "string.h" #include "rtcpabstractserver.-
h"
```

Namespaces

- namespace Coral

7.70 rtcpabstractserver.h File Reference

```
#include "system.h"  #include "timedthread.h"  #include
"rtcppacket.h" #include "internetflow.h" #include "sourcestateinfo.-
h" #include <multimap.h> #include <algo.h> #include "rtcpabstractserver.-
icc"
```

Classes

- class Coral::RTCPAbstractServer

  RTCP abstract server.
- struct Coral::RTCPAbstractServer::Client

Namespaces

- namespace Coral

7.71 rtcppacket.cc File Reference

```
#include "system.h" #include "rtcppacket.h"
```

Namespaces

- namespace Coral

7.72 rtcppacket.h File Reference

```
#include <endian.h> #include "system.h" #include "rtppacket.-
h" #include "rtcppacket.icc"
```

Classes

- class Coral::RTCPCommonHeader

  RTCP Common Header.
- class Coral::RTCPSenderInfoBlock

  RTCP Sender Info Block.

- class Coral::RTCPReceptionReportBlock

  RTCP Reception Report Block.
- class Coral::RTCPReport

  RTCP Report.
- class Coral::RTCPSenderReport

  RTCP Sender Report.
- class Coral::RTCPReceiverReport

  RTCP Sender Report.
- class Coral::RTCPSourceDescriptionItem

  RTCP Source Description Item.
- class Coral::RTCPSourceDescriptionChunk

  RTCP Source Description Chunk.
- class Coral::RTCPSourceDescription

  RTCP Source Description (SDES)
- class Coral::RTCPBye

  RTCP BYE Message.
- class Coral::RTCPApp

  RTCP APP Message.

Namespaces

- namespace Coral

Enumerations

- enum Coral::RTCP_Type { Coral::RTCP_SR = 200, Coral::RTCP_RR = 201, Coral::RTCP_SDES = 202, Coral::RTCP_BYE = 203, Coral::RTCP_APP = 204 }
- enum Coral::RTCP_SDES_Type { Coral::RTCP_SDES_END = 0, Coral::RTCP_SDES_CNAME = 1, Coral::RTCP_SDES_NAME = 2, Coral::RTCP_SDES_EMAIL = 3, Coral::RTCP_SDES_PHONE = 4, Coral::RTCP_SDES_LOC = 5, Coral::RTCP_SDES_TOOL = 6, Coral::RTCP_SDES_NOTE = 7, Coral::RTCP_SDES_PRIV = 8 }

7.73 rtcpreceiver.cc File Reference

```
#include "system.h"  #include "rtcpreceiver.h"  #include "internetflow.h" #include "tools.h"
```

Namespaces

- namespace Coral

Defines

- #define DEBUG

7.73.1 Define Documentation

7.73.1.1 #define DEBUG

7.74 rtcpreceiver.h File Reference

```
#include "system.h" #include "socket.h" #include "thread.-
h" #include "rtcppacket.h" #include "rtcpabstractserver.-
h"
```

Classes

- class Coral::RTCPReceiver

  RTCP Receiver.

Namespaces

- namespace Coral

7.75 rtcpsender.cc File Reference

```
#include "system.h" #include "rtcpsender.h"
```

Namespaces

- namespace Coral

7.76 rtcpsender.h File Reference

```
#include "system.h" #include "socket.h" #include "timedthread.-
h" #include "rtcppacket.h" #include "rtpreceiver.h" #include
"randomizer.h" #include <multimap.h> #include <algo.h>
```

Classes

- class Coral::RTCPSender

  RTCP Sender.

Namespaces

- namespace Coral

7.77 rtppacket.cc File Reference

```
#include "system.h" #include "rtppacket.h"
```

Namespaces

- namespace Coral

Functions

- ostream & Coral::operator<< (ostream &os, const RTPPacket &packet)

7.78 rtppacket.h File Reference

```
#include "system.h" #include "socket.h" #include "rtppacket.-
icc"
```

Classes

- class Coral::RTPPacket

  RTP Packet.

Namespaces

- namespace Coral
- namespace Coral::RTPConstants

Variables

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

7.79 rtpreceiver.cc File Reference

```
#include "system.h"   #include "rtpreceiver.h"   #include
"rtcppacket.h" #include "internetflow.h" #include "randomizer.-
h"
```

Namespaces

- namespace Coral

Defines

- #define DEBUG

7.79.1 Define Documentation

7.79.1.1 #define DEBUG

7.80 rtpreceiver.h File Reference

```
#include "system.h" #include "thread.h" #include "socket.-
h"  #include "rtppacket.h"  #include "decoderinterface.h" ×
#include "sourcestateinfo.h"   #include "internetflow.h" ×
#include "rtpreceiver.icc"
```

Classes

- class Coral::RTPReceiver

  RTP Receiver.

Namespaces

- namespace Coral

7.81 rtpsender.cc File Reference

```
#include "system.h" #include "rtpsender.h" #include "rtcpsender.-
h" #include "randomizer.h" #include <signal.h>
```

Namespaces

- namespace Coral

7.82 rtpsender.h File Reference

```
#include "system.h"   #include "timedthread.h"   #include
"socket.h" #include "rtppacket.h" #include "encoderinterface.-
h" #include "trafficshaper.h" #include "abstractqosdescription.-
h" #include "bandwidthmanager.h" #include "rtpsender.icc"
```

Classes

- class Coral::RTPSender

  RTP Sender.

Namespaces

- namespace Coral

7.83 run.cc File Reference

```
#include "system.h" #include <fstream.h>
```

Functions

- int main (int argc, char *argv[])

7.83.1 Function Documentation

7.83.1.1 int main (int *argc,* char * *argv[]*)

7.84 selectdata.cc File Reference

```
#include "system.h" #include "tools.h" #include "strings.-
h" #include <fstream.h> #include <time.h>
```

Functions

- int main (int argc, char *argv[])

7.84.1 Function Documentation

7.84.1.1 int main (int *argc,* char * *argv[]*)

7.85 seqnumvalidator.cc File Reference

```
#include "system.h" #include "seqnumvalidator.h" #include
"rtppacket.h" #include "tools.h"
```

Namespaces

- namespace Coral

7.86 seqnumvalidator.h File Reference

```
#include "system.h" #include "seqnumvalidator.icc"
```

Classes

- class Coral::SeqNumValidator

  Sequence Number Validator.

Namespaces

- namespace Coral

7.87 servicelevelagreement.cc File Reference

```
#include "system.h" #include "servicelevelagreement.h"
```

Namespaces

- namespace Coral

Functions

- ostream & Coral::operator<< (ostream &os, const ServiceLevelAgreement sla)

7.88 servicelevelagreement.h File Reference

```
#include "system.h"  #include "bandwidthinfo.h"  #include
"trafficclassvalues.h" #include "abstractlayerdescription.-
h" #include "servicelevelagreement.icc"
```

Classes

- struct Coral::DiffServClass

  DiffServ Class.
- class Coral::ServiceLevelAgreement

  Trace Layer Configuration.

Namespaces

- namespace Coral

Functions

- ostream & Coral::operator<< (ostream &os, const ServiceLevelAgreement sla)

7.89 sessiondescription.h File Reference

```
#include "system.h" #include "streamdescription.h" #include
<multimap.h>
```

Classes

- struct Coral::SessionDescription

  Session Description.

Namespaces

- namespace Coral

7.90 simulationgenerator.cc File Reference

```
#include "system.h" #include "tools.h" #include "strings.-
h" #include <fstream.h>
```

Functions

- int main (int argc, char *argv[])

7.90.1 Function Documentation

7.90.1.1 int main (int *argc*, char * *argv[]*)

7.91 socket.cc File Reference

```
#include "system.h" #include "socket.h" #include "randomizer.-
h" #include "in6.h" #include <netdb.h> #include <netinet/in.-
h> #include <netinet/tcp.h> #include <sys/uio.h> #include
<sys/socket.h>
```

Namespaces

- namespace Coral

7.92 socket.h File Reference

```
#include "system.h" #include "internetaddress.h" #include
"internetflow.h" #include "socket.icc"
```

Classes

- class Coral::Socket

  Socket.

Namespaces

- namespace Coral

Variables

- const cardinal Coral::UDPHeaderSize = 8
- const cardinal Coral::IPv4HeaderSize = 20
- const cardinal Coral::IPv6HeaderSize = 40

7.93 socketaddress.h File Reference

```
#include "system.h" #include <sys/socket.h> #include <sys/un.-
h> #include "socketaddress.icc"
```

Classes

- class Coral::SocketAddress

  Socket Address.

Namespaces

- namespace Coral

Functions

- ostream & Coral::operator<< (ostream &os, const SocketAddress &sa)

7.94 sourcestateinfo.cc File Reference

```
#include "sourcestateinfo.h"
```

Namespaces

- namespace Coral

Defines

- #define RTP_SEQ_MOD (1 << 16)
- #define RTP_MAX_DROPOUT 3000
- #define RTP_MAX_MISORDER 100
- #define RTP_MIN_SEQUENTIAL 2

7.94.1 Define Documentation

7.94.1.1 #define RTP_MAX_DROPOUT 3000

7.94.1.2 #define RTP_MAX_MISORDER 100

7.94.1.3 #define RTP_MIN_SEQUENTIAL 2

7.94.1.4 #define RTP_SEQ_MOD (1 << 16)

7.95 sourcestateinfo.h File Reference

```
#include "system.h"  #include "synchronizable.h"  #include
"seqnumvalidator.h" #include "sourcestateinfo.icc"
```

Classes

- class Coral::SourceStateInfo

  Source State Info.

Namespaces

- namespace Coral

7.96 streamdescription.cc File Reference

```
#include "system.h" #include "streamdescription.h"
```

Namespaces

- namespace Coral

7.97 streamdescription.h File Reference

```
#include "system.h"     #include "servicelevelagreement.h"
#include "trafficclassvalues.h" #include "managedstreaminterface.-
h" #include "abstractqosdescription.h" #include "sessiondescription.-
h"
```

Classes

- class Coral::StreamDescription

  Stream Description.

Namespaces

- namespace Coral

7.98 strings.cc File Reference

```
#include "system.h" #include "strings.h" #include <ctype.-
h>
```

Functions

- ostream & operator<< (ostream &os, const String &string)
- String operator+ (const String &string1, const String &string2)

7.98.1 Function Documentation

7.98.1.1 String operator+ (const String & *string1,* const String & *string2*)

Implementation of + operator.

7.98.1.2 ostream& operator<< (ostream & *out,* const String & *string*)

Implementation of << operator.

7.99 strings.h File Reference

```
#include "system.h" #include "strings.icc"
```

Classes

- class String

  String.

Functions

- ostream & operator<< (ostream &out, const String &string)
- String operator+ (const String &string1, const String &string2)

7.99.1 Function Documentation

7.99.1.1 String operator+ (const String & *string1,* const String & *string2*)

Implementation of + operator.

7.99.1.2 ostream& operator<< (ostream & *out,* const String & *string*)

Implementation of << operator.

7.100 synchronizable.cc File Reference

```
#include "system.h"  #include "synchronizable.h"  #include
<pthread.h>
```

Namespaces

- namespace Coral

7.101 synchronizable.h File Reference

```
#include "system.h" #include <pthread.h> #include "synchronizable.-
icc"
```

Classes

- class Coral::Synchronizable

  Synchronizable.

Namespaces

- namespace Coral

7.102 system.h File Reference

```
#include <stdio.h> #include <stdlib.h> #include <unistd.-
h>  #include <errno.h>  #include <iostream.h>  #include
<string.h> #include <math.h> #include <endian.h> #include
<ctype.h> #include "tools.h"
```

Defines

- #define USE_TRAFFICSHAPER
- #define _THREAD_SAFE
- #define _GNU_SOURCE
- #define USE_PTHREADS
- #define CPU_BYTEORDER __BYTE_ORDER

Typedefs

- typedef signed char sbyte
- typedef unsigned char ubyte
- typedef signed char int8
- typedef unsigned char card8
- typedef signed short int16
- typedef unsigned short card16
- typedef signed int int32
- typedef signed int integer
- typedef unsigned int card32
- typedef signed long long int64
- typedef unsigned long long card64
- typedef unsigned int cardinal

7.102.1 Define Documentation

7.102.1.1 #define _GNU_SOURCE

7.102.1.2 #define _THREAD_SAFE

7.102.1.3 #define CPU_BYTEORDER __BYTE_ORDER

7.102.1.4 #define USE_PTHREADS

7.102.1.5 #define USE_TRAFFICSHAPER

7.102.2 Typedef Documentation

7.102.2.1 typedef unsigned short card16

Datatype for storing a 16-bit cardinal.

7.102.2.2 typedef unsigned int card32

Datatype for storing a 32-bit cardinal.

7.102.2.3 typedef unsigned long long card64

Datatype for storing a 64-bit cardinal.

7.102.2.4 typedef unsigned char card8

Datatype for storing a 8-bit cardinal.

7.102.2.5 typedef unsigned int cardinal

Datatype for storing a default-sized cardinal (32 bits minimum).

7.102.2.6 typedef signed short int16

Datatype for storing a 16-bit integer.

7.102.2.7 typedef signed int int32

Datatype for storing a 32-bit intger.

7.102.2.8 typedef signed long long int64

Datatype for storing an 64-bit integer.

7.102.2.9 typedef signed char int8

Datatype for storing an 8-bit integer.

7.102.2.10 typedef signed int integer

Datatype for storing a default-sized integer (32 bits minimum).

7.102.2.11 typedef signed char sbyte

Datatype for storing a signed char.

7.102.2.12 typedef unsigned char ubyte

Datatype for storing an unsigned char.

7.103 t0.cc File Reference

`#include "system.h" #include "tdtf.h" #include "tdtfmediareader.h"`

Functions

- int main (int argc, char *argv[])

7.103.1 Function Documentation

7.103.1.1 int main (int *argc,* char * *argv[]*)

7.104 t1.cc File Reference

`#include "system.h" #include "tdtf.h" #include "tdtfreader.h"`

Functions

- int main (int argc, char *argv[])

7.104.1 Function Documentation

7.104.1.1 int main (int *argc,* char * *argv[]*)

7.105 t2.cc File Reference

`#include "system.h" #include "tdtf.h" #include "tdtfmediareader.h"`

Functions

- void test (TDTFMediaReader *t)
- int main (int argc, char *argv[])

Variables

- int TEST = 1

7.105.1 Function Documentation

7.105.1.1 int main (int *argc,* char * *argv[]*)

7.105.1.2 void test (TDTFMediaReader * *t*)

7.105.2 Variable Documentation

7.105.2.1 int TEST = 1

7.106 t3.cc File Reference

```
#include "system.h" #include <fstream.h>
```

Functions

- int main (int argc, char *argv[])

7.106.1 Function Documentation

7.106.1.1 int main (int *argc*, char * *argv[]*)

7.107 t4.cc File Reference

```
#include "system.h" #include "tdtf.h" #include "tracearray.-
h"  #include "mpegtracearray.h"  #include "tdtfwriter.h" ×
#include "tdtfreader.h" #include <sys/time.h>
```

Functions

- int main (int argc, char *argv[])

Variables

- TraceConfiguration gConfig

7.107.1 Function Documentation

7.107.1.1 int main (int *argc*, char * *argv[]*)

7.107.2 Variable Documentation

7.107.2.1 TraceConfiguration gConfig

7.108 t5.cc File Reference

```
#include "system.h" #include "socket.h" #include "internetaddress.-
h"  #include "timedthread.h"  #include "breakdetector.h" ×
#include "trafficclassvalues.h"  #include "trafficshaper.-
h"  #include "trafficpolicer.h"  #include <multimap.h> ×
#include <vector.h>
```

Classes

- class SenderThread

Functions

- int main (int argc, char **argv)

7.108.1 Function Documentation

7.108.1.1 int main (int *argc,* char ** *argv*)

7.109 t6.cc File Reference

`#include "system.h" #include "tools.h" #include <sys/mman.-h> #include <fcntl.h> #include <limits.h>`

Functions

- cardinal getBit (const char *data, const cardinal bit)
- cardinal getBlock (const char *data, const cardinal bit, const cardinal length)
- void getTrace (const char *data, const cardinal length)
- int main (int argc, char **argv)

Variables

- const char * Format [8]

7.109.1 Function Documentation

7.109.1.1 cardinal getBit (const char * *data,* const cardinal *bit*) [inline]

7.109.1.2 cardinal getBlock (const char * *data,* const cardinal *bit,* const cardinal *length*) [inline]

7.109.1.3 void getTrace (const char * *data,* const cardinal *length*)

7.109.1.4 int main (int *argc,* char ** *argv*)

7.109.2 Variable Documentation

7.109.2.1 const char* Format[8]

Initial value:

```
{
   "", "sub-QCIF", "QCIF", "CIF", "4CIF", "16CIF", "?", "extended PTYPE"
}
```

7.110 t7.cc File Reference

`#include "system.h" #include "tools.h" #include "strings.-h" #include <fstream.h> #include <time.h>`

Classes

- struct MediaList

Functions

- int main (int argc, char ∗argv[])

Variables

- const cardinal Entries = 100
- MediaList MList [Entries]

7.110.1 Function Documentation

7.110.1.1 int main (int *argc,* char ∗ *argv[]*)

7.110.2 Variable Documentation

7.110.2.1 const cardinal Entries = 100

7.110.2.2 MediaList MList[Entries]

7.111 t8.cc File Reference

`#include "system.h" #include "tools.h" #include "strings.-h"`

Functions

- int main (int argc, char ∗argv[])

7.111.1 Function Documentation

7.111.1.1 int main (int *argc,* char * *argv[]*)

7.112 tclient.cc File Reference

```
#include "socket.h"   #include "traceclient.h"   #include
"breakdetector.h" #include "strings.h" #include <fstream.-
h>
```

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

- TraceClient * client = NULL
- ofstream * gpScript = NULL
- ofstream * gpData = NULL

7.112.1 Function Documentation

7.112.1.1 void cleanUp (const cardinal *exitCode* = 0)

7.112.1.2 void initGNUplot (const char * *prefix,* const char * *info,* const cardinal *layers,* const String & *address*)

7.112.1.3 int main (int *argc,* char * *argv[]*)

7.112.2 Variable Documentation

7.112.2.1 TraceClient* client = NULL

7.112.2.2 ofstream* gpData = NULL

7.112.2.3 ofstream* gpScript = NULL

7.113 tdtf.cc File Reference

```
#include "system.h" #include "tdtf.h"
```

Namespaces

- namespace Coral

7.114 tdtf.h File Reference

```
#include "system.h" #include "tdtf.icc"
```

Classes

- struct Coral::TDTFPrefixExtensionHeader

  TDTF Prefix Extension Header.
- struct Coral::TDTFPrefix

  TDTF Prefix.
- struct Coral::TDTFSuffix

  TDTF Suffix.
- struct Coral::EmpiricalEnvelopePair

  Empirical Envelope Pair.
- struct Coral::EmpiricalEnvelope

  Empirical Envelope Header.
- struct Coral::FrameDescription

  Frame Description.
- struct Coral::TraceHeader

  Trace Header.
- struct Coral::UtilizationHeader

  Trace Header.
- struct Coral::ResourceUtilizationEntry

  Resource Utilization Entry.
- struct Coral::ResourceUtilizationHeader

  Resource Utilization Header.
- struct Coral::IntervalHeader

  Interval Header.
- struct Coral::LayerHeader

  Layer Header.
- struct Coral::PositionLengthIntervalIndexEntry

  Layer Header.
- struct Coral::PositionLengthIntervalIndexHeader

  Layer Header.
- struct Coral::ResourceUtilizationListIndexEntry

  Resource Utilization List Index Entry.
- struct Coral::ResourceUtilizationListIndexHeader

  Resource Utilization List Index Header.

- struct Coral::MainIndexEntry

  Layer Header.
- struct Coral::MainIndexHeader

  Main Index Header.

Namespaces

- namespace Coral

Enumerations

- enum Coral::UtilityFunctions { Coral::UF_Linear = 0x0000, Coral::UF_-Exponential1 = 0x0010, Coral::UF_Exponential2 = 0x0011, Coral::UF_Undefined = 0xffff }

7.115 tdtfmediareader.cc File Reference

```
#include "system.h" #include "tdtfmediareader.h"
```

Namespaces

- namespace Coral

7.116 tdtfmediareader.h File Reference

```
#include "system.h" #include "tdtfreader.h" #include "mediainfo.-
h" #include "tdtfmediareader.icc"
```

Classes

- class Coral::TDTFMediaReader

  TDTF Media Reader.

Namespaces

- namespace Coral

7.117 tdtfreader.cc File Reference

```
#include "system.h" #include "tdtfreader.h" #include "synchronizable.-
h"  #include <sys/mman.h>  #include <fcntl.h>  #include
<limits.h> #include "algo.h"
```

Namespaces

- namespace Coral

7.118 tdtfreader.h File Reference

```
#include "system.h" #include "tdtf.h" #include "synchronizable.-
h" #include <multimap.h> #include "tdtfreader.icc"
```

Classes

- class Coral::TDTFReader

  Trace Reader.
- struct Coral::TDTFReader::MediaCacheEntry

Namespaces

- namespace Coral

7.119 tdtfwriter.cc File Reference

```
#include "system.h" #include "tdtfwriter.h" #include "tools.-
h"
```

Namespaces

- namespace Coral

7.120 tdtfwriter.h File Reference

```
#include "system.h" #include "tracearray.h" #include "traceconfiguration.-
h"
```

Classes

- class Coral::TDTFWriter

  TDTF Writer.

Namespaces

- namespace Coral

7.121 tgenerator.cc File Reference

```
#include "system.h" #include "tdtf.h" #include "mpegtracearray.-
h" #include "h263tracearray.h" #include "mp3tracearray.h"
#include "globaltraceconfiguration.h" #include "tdtfwriter.-
h"
```

Functions

- int main (int argc, char *argv[])

7.121.1 Function Documentation

7.121.1.1 int main (int *argc,* char * *argv[]*)

7.122 thread.cc File Reference

```
#include "system.h" #include "thread.h" #include <sys/time.-
h>
```

Namespaces

- namespace Coral

7.123 thread.h File Reference

```
#include "system.h" #include "synchronizable.h" #include
<pthread.h> #include "thread.icc"
```

Classes

- class Coral::Thread

  Thread.

Namespaces

- namespace Coral

7.124 timedthread.cc File Reference

```
#include "system.h"   #include "timedthread.h"   #include
"tools.h" #include <sys/time.h> #include <signal.h>
```

Namespaces

- namespace Coral

7.125 timedthread.h File Reference

```
#include "system.h" #include "thread.h" #include <signal.-
h> #include "timedthread.icc"
```

Classes

- class Coral::TimedThread

  Timed Thread.

Namespaces

- namespace Coral

7.126 tools.cc File Reference

```
#include <sys/time.h> #include "system.h" #include "strings.-
h" #include "tools.h"
```

Namespaces

- namespace Coral

Functions

- card64 Coral::getMicroTime ()
- cardinal Coral::calculatePacketsPerSecond (const cardinal payloadBytesPerSecond, const cardinal framesPerSecond, const cardinal maxPacketSize, const cardinal headerLength)
- cardinal Coral::calculateBytesPerSecond (const cardinal payloadBytesPerSecond, const cardinal framesPerSecond, const cardinal maxPacketSize, const cardinal headerLength)
- bool Coral::scanURL (const String &location, String &protocol, String &host, String &path)
- void Coral::printTimeStamp (ostream &os)

7.127 tools.h File Reference

```
#include "system.h" #include "strings.h" #include "tools.icc"
```

Namespaces

- namespace Coral

Functions

- void Coral::debug (const char *string)
- card64 Coral::getMicroTime ()
- card16 Coral::translate16 (const card16 x)
- card32 Coral::translate32 (const card32 x)
- card64 Coral::translate64 (const card64 x)
- card64 Coral::translateToBinary (const double x)
- double Coral::translateToDouble (const card64 x)
- cardinal Coral::calculatePacketsPerSecond (const cardinal payloadBytesPerSecond, const cardinal framesPerSecond, const cardinal maxPacketSize, const cardinal headerLength)
- cardinal Coral::calculateBytesPerSecond (const cardinal payloadBytesPerSecond, const cardinal framesPerSecond, const cardinal maxPacketSize, const cardinal headerLength)
- bool Coral::scanURL (const String &location, String &protocol, String &host, String &path)
- template<class T >
  void Coral::quickSort (T *array, const integer start, const integer end)
- template<class T >
  cardinal Coral::removeDuplicates (T *array, const cardinal length)
- void Coral::printTimeStamp (ostream &os)

7.128 totalanalyzer.cc File Reference

```
#include "system.h" #include "tools.h" #include "strings.-
h" #include <fstream.h> #include <time.h>
```

Functions

- int main (int argc, char *argv[])

7.128.1 Function Documentation

7.128.1.1 int main (int *argc*, char * *argv[]*)

7.129 tprinter.cc File Reference

```
#include "system.h" #include "tdtf.h" #include "tdtfreader.-
h" #include <sys/time.h>
```

Functions

- int main (int argc, char *argv[])

7.129.1 Function Documentation

7.129.1.1 int main (int *argc*, char * *argv[]*)

7.130 tracearray.cc File Reference

```
#include "system.h" #include "tracearray.h"
```

Namespaces

- namespace Coral

Functions

- ostream & Coral::operator<< (ostream &os, const TraceArray &traceArray)

7.131 tracearray.h File Reference

```
#include "system.h" #include "traceconfiguration.h" #include
"tracearray.icc"
```

Classes

- class Coral::TraceArray

  Trace Array.
- struct Coral::TraceArray::Trace

Namespaces

- namespace Coral

Functions

- ostream & Coral::operator<< (ostream &os, const TraceArray &traceArray)

7.132 traceclient.cc File Reference

```
#include "socket.h"   #include "tracedecoder.h"   #include
"tracedecoderrepository.h" #include "rtpreceiver.h" #include
"rtcppacket.h"   #include "rtcpsender.h"   #include "set.h"
#include "tools.h" #include "strings.h" #include "randomizer.-
h" #include "traceclientapppacket.h" #include "traceclient.-
h"
```

Namespaces

- namespace Coral

7.133 traceclient.h File Reference

```
#include "tracedecoderinterface.h" #include "tracedecoderrepository.-
h" #include "mediainfo.h" #include "rtcpsender.h" #include
"rtpreceiver.h"    #include "internetaddress.h"    #include
"socket.h" #include "strings.h" #include "traceclientapppacket.-
h" #include <multimap.h> #include <algo.h> #include "traceclient.-
icc"
```

Classes

- class Coral::TraceClient
  Trace Client.

Namespaces

- namespace Coral

7.134 traceclientapppacket.cc File Reference

```
#include "system.h"    #include "traceclientapppacket.h" ×
#include "tools.h"
```

Namespaces

- namespace Coral

7.135 traceclientapppacket.h File Reference

Classes

- class Coral::TraceClientAppPacket
  Trace Client RTCP-SDES-APP-PRIV Packet.

Namespaces

- namespace Coral

Variables

- const card8 Coral::TraceServerDefaultTrafficClass = 0x00
- const card8 Coral::TraceClientDefaultTrafficClass = 0x00

7.136 traceconfiguration.cc File Reference

```
#include "system.h" #include "traceconfiguration.h"
```

Namespaces

- namespace Coral

Functions

- ostream & Coral::operator<< (ostream &os, const TraceConfiguration &config)

7.137 traceconfiguration.h File Reference

```
#include "system.h" #include "tdtf.h"
```

Classes

- struct Coral::TraceLayerConfiguration

  Trace Layer Configuration.
- struct Coral::TraceConfiguration

  Trace Configuration.

Namespaces

- namespace Coral

Functions

- ostream & Coral::operator<< (ostream &os, const TraceConfiguration &config)

7.138 tracedecoder.cc File Reference

```
#include "system.h"   #include "tracedecoder.h"   #include
"tracepacket.h"   #include "seqnumvalidator.h"   #include
"tools.h"
```

Namespaces

- namespace Coral

7.139 tracedecoder.h File Reference

```
#include "system.h"  #include "synchronizable.h"  #include
"tracedecoderinterface.h"   #include "seqnumvalidator.h" ×
#include "rtppacket.h"
```

Classes

- class Coral::TraceDecoder

  Trace Decoder.

Namespaces

- namespace Coral

7.140 tracedecoderinterface.h File Reference

```
#include "system.h" #include "decoderinterface.h" #include
"range.h"
```

Classes

- class Coral::TraceDecoderInterface

  Trace Decoder Interface.

Namespaces

- namespace Coral

7.141 tracedecoderrepository.cc File Reference

```
#include "system.h" #include "tracedecoderrepository.h"
```

Namespaces

- namespace Coral

7.142 tracedecoderrepository.h File Reference

```
#include "system.h"     #include "tracedecoderinterface.h"
#include "decoderrepositoryinterface.h" #include <multimap.-
h>  #include <algo.h>  #include "tracedecoderrepository.-
icc"
```

Classes

- class Coral::TraceDecoderRepository
  Trace Decoder Repository.

Namespaces

- namespace Coral

7.143 traceencoder.cc File Reference

```
#include "system.h"   #include "traceencoder.h"   #include
"tracepacket.h" #include "mediainfo.h" #include "tools.h"
#include "traceqosdescription.h"
```

Classes

- struct Coral::QualityScenarioEntry
- struct Coral::QualityScenario

Namespaces

- namespace Coral

Variables

- QualityScenario Coral::QualityScenarios []

7.144 traceencoder.h File Reference

```
#include "system.h"     #include "traceencoderinterface.h"
#include "tdtfmediareader.h" #include "abstractqosdescription.-
h" #include "rtppacket.h"
```

Classes

- class Coral::TraceEncoder
  Trace Encoder.

Namespaces

- namespace Coral

Defines

- #define TRACEENCDOER_H

7.144.1 Define Documentation

7.144.1.1 #define TRACEENCDOER_H

7.145 traceencoderinterface.h File Reference

```
#include "system.h" #include "encoderinterface.h" #include
"mediainfo.h"
```

Classes

- class Coral::TraceEncoderInterface

  Trace Encoder Interface.

Namespaces

- namespace Coral

7.146 traceencoderrepository.cc File Reference

```
#include "system.h" #include "traceencoderrepository.h"
```

Namespaces

- namespace Coral

7.147 traceencoderrepository.h File Reference

```
#include "system.h"  #include "synchronizable.h"  #include
"encoderrepositoryinterface.h" #include "traceencoderinterface.-
h" #include <multimap.h> #include <algo.h> #include "traceencoderrepository
icc"
```

Classes

- class Coral::TraceEncoderRepository

  Trace Encoder Repository.

Namespaces

- namespace Coral

7.148 traceframeratescalability.cc File Reference

```
#include "system.h" #include "traceframeratescalability.-
h" #include "utilityfunction.h"
```

Namespaces

- namespace Coral

7.149 traceframeratescalability.h File Reference

```
#include "system.h" #include "frameratescalabilityinterface.-
h" #include "tdtfreader.h"
```

Classes

- class Coral::TraceFrameRateScalability

  Trace Frame Rate Scalability.

Namespaces

- namespace Coral

7.150 traceframesizescalability.cc File Reference

```
#include "system.h" #include "traceframesizescalability.-
h" #include "utilityfunction.h"
```

Namespaces

- namespace Coral

7.151 traceframesizescalability.h File Reference

```
#include "system.h" #include "framesizescalabilityinterface.-
h" #include "genericframesizescalability.h" #include "tdtfreader.-
h"
```

Classes

- class Coral::TraceFrameSizeScalability

  Trace Frame Size Scalability.

Namespaces

- namespace Coral

7.152 tracepacket.cc File Reference

```
#include "system.h"   #include "tracepacket.h"   #include
"tools.h"
```

Namespaces

- namespace Coral

7.153 tracepacket.h File Reference

```
#include "system.h" #include "mediainfo.h"
```

Classes

- class Coral::TracePacket

  Trace Packet.
- struct Coral::TracePacketData

  Trace Packet Data.

Namespaces

- namespace Coral

7.154 traceqosdescription.cc File Reference

```
#include "system.h" #include "traceqosdescription.h"
```

Namespaces

- namespace Coral

7.155 traceqosdescription.h File Reference

```
#include "system.h"    #include "abstractqosdescription.h"
#include "traceframeratescalability.h" #include "traceframesizescalability.-
h"
```

Classes

- class Coral::TraceLayerDescription

  Trace Layer QoS Description.
- class Coral::TraceQoSDescription

  Trace QoS Description.

Namespaces

- namespace Coral

7.156 traceserver.cc File Reference

```
#include "system.h" #include "tdtfreader.h" #include "socket.-
h" #include "traceencoder.h" #include "traceencoderrepository.-
h" #include "rtpsender.h" #include "rtcppacket.h" #include
"rtcpreceiver.h" #include "rtcpabstractserver.h" #include
"mediainfo.h" #include "sourcestateinfo.h" #include "tools.-
h" #include "randomizer.h" #include "traceclientapppacket.-
h" #include "traceserver.h"
```

Namespaces

- namespace Coral

Defines

- #define VERBOSE

7.156.1 Define Documentation

7.156.1.1 #define VERBOSE

7.157 traceserver.h File Reference

```
#include "system.h" #include "tdtfmediareader.h" #include
"socket.h"  #include "traceencoderrepository.h"  #include
"rtpsender.h" #include "rtcppacket.h" #include "rtcpreceiver.-
h" #include "rtcpabstractserver.h" #include "traceclientapppacket.-
h"  #include "bandwidthmanager.h"  #include <multimap.h> ×
#include <algo.h> #include "traceserver.icc"
```

Classes

- class Coral::TraceServer

  Trace Server.
- struct Coral::TraceServer::User

Namespaces

- namespace Coral

7.158 trafficclassvalues.cc File Reference

```
#include "system.h" #include "trafficclassvalues.h"
```

Namespaces

- namespace Coral

7.159 trafficclassvalues.h File Reference

```
#include "system.h" #include "trafficclassvalues.icc"
```

Classes

- class Coral::TrafficClassValues

  Traffic Class Values.

Namespaces

- namespace Coral

7.160 trafficpolicer.cc File Reference

```
#include "system.h"  #include "trafficpolicer.h"  #include
"tools.h"
```

Namespaces

- namespace Coral

7.161 trafficpolicer.h File Reference

```
#include "system.h" #include <multimap.h> #include "trafficpolicer.-
icc"
```

Classes

- class Coral::TrafficPolicer

  Traffic Policer.
- struct Coral::TrafficPolicer::TrafficPolicerPacket

Namespaces

- namespace Coral

7.162 trafficshaper.cc File Reference

```
#include "system.h"  #include "trafficshaper.h"  #include
"tools.h" #include <algo.h>
```

Namespaces

- namespace Coral

7.163 trafficshaper.h File Reference

```
#include "system.h" #include "socket.h" #include "internetaddress.-
h" #include "timedthread.h" #include "trafficclassvalues.-
h" #include <deque.h> #include <vector.h> #include "trafficshaper.-
icc"
```

Classes

- class Coral::TrafficShaperSingleton

  Traffic Shaper Singleton.
- class Coral::TrafficShaper

  Traffic Shaper.
- struct Coral::TrafficShaper::TrafficShaperPacket

Namespaces

- namespace Coral

7.164 tserver.cc File Reference

```
#include "system.h" #include "socket.h" #include "rtpsender.-
h"   #include "rtcppacket.h"   #include "rtcpreceiver.h" ×
#include "rtcpabstractserver.h" #include "traceclientapppacket.-
h" #include "tools.h" #include "breakdetector.h" #include
"traceserver.h"   #include "bandwidthmanager.h"   #include
"servicelevelagreement.h" #include "roundtriptimepinger.-
h"
```

Functions

- void cleanUp (const cardinal exitCode=0)
- void initAll (const char *directory, const card16 port, const card64 timeout, const cardinal maxPacketSize, const bool lossScalability)
- int main (int argc, char *argv[])

Variables

- Socket * rtcpServerSocket = NULL
- InternetAddress * rtcpServerAddress = NULL
- RTCPReceiver * rtcpReceiver = NULL
- TraceServer * server = NULL
- ServiceLevelAgreement * sla = NULL

- BandwidthManager * bwManager = NULL
- Socket * pingSocket4 = NULL
- Socket * pingSocket6 = NULL
- RoundTripTimePinger * pinger = NULL
- ofstream * logStream = NULL

7.164.1 Function Documentation

7.164.1.1 void cleanUp (const cardinal *exitCode* = 0)

7.164.1.2 void initAll (const char * *directory*, const card16 *port*, const card64 *timeout*, const cardinal *maxPacketSize*, const bool *lossScalability*)

7.164.1.3 int main (int *argc*, char * *argv[]*)

7.164.2 Variable Documentation

7.164.2.1 BandwidthManager* bwManager = NULL

7.164.2.2 ofstream* logStream = NULL

7.164.2.3 RoundTripTimePinger* pinger = NULL

7.164.2.4 Socket* pingSocket4 = NULL

7.164.2.5 Socket* pingSocket6 = NULL

7.164.2.6 RTCPReceiver* rtcpReceiver = NULL

7.164.2.7 InternetAddress* rtcpServerAddress = NULL

7.164.2.8 Socket* rtcpServerSocket = NULL

7.164.2.9 TraceServer* server = NULL

7.164.2.10 ServiceLevelAgreement* sla = NULL

7.165 tsimulator.cc File Reference

```
#include "system.h" #include "rtppacket.h" #include "managedstreaminterface.-
h" #include "trafficclassvalues.h" #include "bandwidthmanager.-
h" #include "tdtfmediareader.h" #include "traceencoder.h"
#include "trafficpolicer.h" #include "strings.h" #include
"gnuplot.h" #include <multimap.h>
```

Classes

- class SimulatorTask

  Simulator Task.
- struct Task
- struct Action

Enumerations

- enum ActionTypes { AT_Bandwidth = 1, AT_TransferDelay = 2, AT_LossRate = 3, AT_Jitter = 4 }

Functions

- Action * newAction (ServiceLevelAgreement *sla, Action *lastAction, const double offset, const char *className, const cardinal type, const card64 bandwidth, const double transferDelay, const double lossRate, const double jitter)
- int main (int argc, char *argv[])

7.165.1 Enumeration Type Documentation

7.165.1.1 enum ActionTypes

Enumerator:

AT_Bandwidth

AT_TransferDelay

AT_LossRate

AT_Jitter

7.165.2 Function Documentation

7.165.2.1 int main (int *argc,* char * *argv[]*)

7.165.2.2 Action* newAction (ServiceLevelAgreement * *sla,* Action * *lastAction,* const double *offset,* const char * *className,* const cardinal *type,* const card64 *bandwidth,* const double *transferDelay,* const double *lossRate,* const double *jitter*)

7.166 tstats.cc File Reference

```
#include "system.h" #include "tdtf.h" #include "tdtfreader.-
h"
```

Functions

- int main (int argc, char ∗argv[])

7.166.1 Function Documentation

7.166.1.1 int main (int *argc,* char ∗ *argv[]*)

7.167 tutilizer.cc File Reference

```
#include "system.h" #include "tdtf.h" #include "tdtfmediareader.-
h" #include "mpegwriterqosdescription.h" #include "h263writerqosdescription.-
h" #include "mp3writerqosdescription.h" #include "globaltraceconfiguration.-
h"
```

Classes

- class TDTFUtilizationUpdater

  TDTF Media Reader.

Functions

- bool copy (const char ∗input, const char ∗output)
- int main (int argc, char ∗argv[])

7.167.1 Function Documentation

7.167.1.1 bool copy (const char ∗ *input,* const char ∗ *output*)

7.167.1.2 int main (int *argc,* char ∗ *argv[]*)

7.168 unixaddress.cc File Reference

```
#include "system.h" #include "strings.h" #include "unixaddress.-
h"
```

Namespaces

- namespace Coral

7.169 unixaddress.h File Reference

`#include "system.h" #include "strings.h" #include "socketaddress.h" #include "portableaddress.h" #include <sys/socket.h> #include <sys/un.h> #include "unixaddress.icc"`

Classes

- class Coral::UnixAddress

  Socket Address.

Namespaces

- namespace Coral

7.170 utilityfunction.cc File Reference

`#include "system.h" #include "utilityfunction.h"`

Namespaces

- namespace Coral

Functions

- double Coral::evaluateUtilityFunction (const cardinal type, const double scaleFactor, const double *constantArray, const cardinal constants)

7.171 utilityfunction.h File Reference

`#include "system.h" #include "tdtf.h" #include "utilityfunction.icc"`

Namespaces

- namespace Coral

Functions

- double Coral::evaluateUtilityFunction (const cardinal type, const double scaleFactor, const double *constantArray, const cardinal constants)
- double Coral::evaluateUtilityFunctionTranslated (const cardinal type, const double scaleFactor, const card64 *constantArray, const cardinal constants)

7.172 vtclient.cc File Reference

```
#include "system.h"   #include "traceclient.h"   #include
"randomizer.h" #include "thread.h" #include "mediainfo.h"
#include "breakdetector.h"
```

Classes

- class VerificationClientThread

Functions

- void validatePr (double &p)
- int main (int argc, char **argv)

Variables

- const cardinal DefaultPause = 500
- const cardinal DefaultThreads = 12
- char * DefaultServer = "localhost:7100"
- char * DefaultMedia = "Test%d.tdtf"
- const cardinal DefaultMediaCount = 1

7.172.1 Function Documentation

7.172.1.1 int main (int *argc,* char ** *argv*)

7.172.1.2 void validatePr (double & *p*) [inline]

7.172.2 Variable Documentation

7.172.2.1 char* DefaultMedia = "Test%d.tdtf"

7.172.2.2 const cardinal DefaultMediaCount = 1

7.172.2.3 const cardinal DefaultPause = 500

7.172.2.4 char* DefaultServer = "localhost:7100"

7.172.2.5 const cardinal DefaultThreads = 12