QSense Sidecar Library

1.0.0

Generated by Doxygen 1.8.9.1

Sun May 24 2015 21:43:54

Contents

Chapter 1

QSense Sidecar Library

1.1 Contents

1.2 Overview

A standard C++ library for interacting with the Sidecar Event API. Developed for use on Arduino platform, but may be easily extended to work on other platforms.

1.3 Workflow

The general workflow for publishing an event to the Sidecar Event API is as follows:

- Initialise the QSense Sidecar Library (Initialisation).
- Create an instance of qsense::Event.
- Add instances of qsense::Reading to the event.
- Add any tag values to the event (tags are instances of qsense::QString).
- Create an instance of qsense::net::SidecarClient
- Publish the event using qsense::net::SidecarClient::publish

1.4 Initialisation

The QSense Sidecar Library needs some initialisation (from the Arduino sketch for instance). The accompanying sketch illustrates the recommended way of initialising the library.

- Initialise the library with the type of networking used by the device. Use the qsense::net::initNetworkType(qsense::net::NetworkType) function to indicate the type of network being used.

- Initialise the UUID engine with a proper MAC address (qsense::UUID::init). Official Arduino ethernet shields come with a MAC address labelled on the board. If using WiFi, the Arduino WiFi API allows lookup of the current MAC address. A stable and unique MAC address is essential for ensuring that UUID values generated (generates time based values) are truly universally unique.
- Initialise the Event API with the stream identifier, device UUID, and a default geographic qsense::Location (qsense::Event::init).
- Initialise the Sidecar library with the api key and secret used for authentication/authorisation (qsense::net::↩ SidecarClient::init).

1.5 Libraries

The only third-party library necessary is the Standard C++ Library.

- `Standard C++ Template Unit`

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

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 qsense Namespace Reference

Namespaces

- hash
- net

Classes

- class ByteOrder

  This class contains a number of static methods to convert between big-endian and little-endian integers of various sizes.
- class Event

  A simple class that encapsulates an event sent to Sidecar. Events are holders for readings. Events can be serialised to JSON using the toString method.
- class Location

  A simple representation of geographical location.
- class Reading

  A class that represents a single reading. Readings are added to an Event.
- class UUID

  A class that represents a UUID/GUID.

Typedefs

- typedef std::string QString
- typedef unsigned char Byte

Functions

- std::ostream & operator<< (std::ostream &os, const Event &event)

  Serialise the specified event as JSON to the output stream.
- std::ostream & operator<< (std::ostream &os, const qsense::Location &location)

  Serialise the specified location to the output stream.
- std::ostream & operator<< (std::ostream &os, const qsense::Reading &reading)

  Serialise the reading as JSON to the output stream.

5.1.1 Detailed Description

The namespace for the QSense Sidecar Library.

5.1.2 Typedef Documentation

5.1.2.1 typedef unsigned char qsense::Byte

8-bit unsigned char

5.1.2.2 typedef std::string qsense::QString

Not really necessary any more. Initially had it to use String class from Arduino library, but that never seems to work when used for hashing etc.

5.1.3 Function Documentation

5.1.3.1 std::ostream& qsense::operator<< (std::ostream & *os,* const qsense::Location & *location*)

Serialise the specified location to the output stream.

5.1.3.2 std::ostream& qsense::operator<< (std::ostream & *os,* const qsense::Reading & *reading*)

Serialise the reading as JSON to the output stream.

5.1.3.3 std::ostream& qsense::operator<< (std::ostream & *os,* const Event & *event*)

Serialise the specified event as JSON to the output stream.

5.2 qsense::hash Namespace Reference

Namespaces

- base64

Classes

- class MD5

  Class for generating MD5 hashes.
- class Sha1

  Class for hashing using SHA1 algorithm.

5.2.1 Detailed Description

Namespace for classes and functions that provide hashing support.

5.3 qsense::hash::base64 Namespace Reference

Functions

- int32_t encodeLength (int32_t len)
- int32_t encode (char *output, const char *input, int32_t inputLength)
- int32_t decodeLength (const char *code)
- int32_t decode (char *outputPlainText, const char *encoded)

5.3.1 Detailed Description

Namespace for functions that provide Base64 encoding/decoding support.

5.3.2 Function Documentation

5.3.2.1 int32_t qsense::hash::base64::decode (char * *outputPlainText*, const char * *encoded*)

Decode into outputPlainText the encoded contents

5.3.2.2 int32_t qsense::hash::base64::decodeLength (const char * *code*)

Use to specify size of output array to decode into

5.3.2.3 int32_t qsense::hash::base64::encode (char * *output*, const char * *input*, int32_t *inputLength*)

Encode into output contents of plain_src of specified length

5.3.2.4 int32_t qsense::hash::base64::encodeLength (int32_t *len*)

Use to specify size of output array to encode into

5.4 qsense::net Namespace Reference

Classes

- class DateTime

  Represents current date/time. Seeds initially (and daily) from a network time service, and uses internal timer to represent a real-time clock.
- class HttpClient

  A HTTP Client class for use with either ethernet or wifi. Before use, the client should be initialised with the network type used by the device (qsense::net::initNetworkType.
- class SidecarClient

  Class that encapsulates interactions with the Sidecar REST API.

Enumerations

- enum NetworkType { Ethernet = 0, WiFi = 1 }

  Enumeration of network connection types for device.

Functions

- void initNetworkType (NetworkType type)

5.4.1 Detailed Description

Namespace for classes that provide network services and require a network connection to work.

5.4.2 Enumeration Type Documentation

5.4.2.1 enum qsense::net::NetworkType

Enumeration of network connection types for device.

Enumerator

Ethernet

WiFi

5.4.3 Function Documentation

5.4.3.1 void qsense::net::initNetworkType (NetworkType *type*)

Initialise the API to use the specified type. HttpClient::create uses this type to create appropriate implementation.

Chapter 6

Class Documentation

6.1 qsense::ByteOrder Class Reference

This class contains a number of static methods to convert between big-endian and little-endian integers of various sizes.

`#include <ByteOrder.h>`

Static Public Member Functions

- static int16_t flipBytes (int16_t value)
- static uint16_t flipBytes (uint16_t value)
- static int32_t flipBytes (int32_t value)
- static uint32_t flipBytes (uint32_t value)
- static int64_t flipBytes (int64_t value)
- static uint64_t flipBytes (uint64_t value)
- static int16_t toBigEndian (int16_t value)
- static uint16_t toBigEndian (uint16_t value)
- static int32_t toBigEndian (int32_t value)
- static uint32_t toBigEndian (uint32_t value)
- static int64_t toBigEndian (int64_t value)
- static uint64_t toBigEndian (uint64_t value)
- static int16_t fromBigEndian (int16_t value)
- static uint16_t fromBigEndian (uint16_t value)
- static int32_t fromBigEndian (int32_t value)
- static uint32_t fromBigEndian (uint32_t value)
- static int64_t fromBigEndian (int64_t value)
- static uint64_t fromBigEndian (uint64_t value)
- static int16_t toLittleEndian (int16_t value)
- static uint16_t toLittleEndian (uint16_t value)
- static int32_t toLittleEndian (int32_t value)
- static uint32_t toLittleEndian (uint32_t value)
- static int64_t toLittleEndian (int64_t value)
- static uint64_t toLittleEndian (uint64_t value)
- static int16_t fromLittleEndian (int16_t value)
- static uint16_t fromLittleEndian (uint16_t value)
- static int32_t fromLittleEndian (int32_t value)
- static uint32_t fromLittleEndian (uint32_t value)
- static int64_t fromLittleEndian (int64_t value)
- static uint64_t fromLittleEndian (uint64_t value)

- static int16_t toNetwork (int16_t value)
- static uint16_t toNetwork (uint16_t value)
- static int32_t toNetwork (int32_t value)
- static uint32_t toNetwork (uint32_t value)
- static int64_t toNetwork (int64_t value)
- static uint64_t toNetwork (uint64_t value)
- static int16_t fromNetwork (int16_t value)
- static uint16_t fromNetwork (uint16_t value)
- static int32_t fromNetwork (int32_t value)
- static uint32_t fromNetwork (uint32_t value)
- static int64_t fromNetwork (int64_t value)
- static uint64_t fromNetwork (uint64_t value)

6.1.1 Detailed Description

This class contains a number of static methods to convert between big-endian and little-endian integers of various sizes.

6.1.2 Member Function Documentation

6.1.2.1 int16_t qsense::ByteOrder::flipBytes (int16_t *value*) [inline], [static]

6.1.2.2 uint16_t qsense::ByteOrder::flipBytes (uint16_t *value*) [inline], [static]

6.1.2.3 int32_t qsense::ByteOrder::flipBytes (int32_t *value*) [inline], [static]

6.1.2.4 uint32_t qsense::ByteOrder::flipBytes (uint32_t *value*) [inline], [static]

6.1.2.5 int64_t qsense::ByteOrder::flipBytes (int64_t *value*) [inline], [static]

6.1.2.6 uint64_t qsense::ByteOrder::flipBytes (uint64_t *value*) [inline], [static]

6.1.2.7 static int16_t qsense::ByteOrder::fromBigEndian (int16_t *value*) [static]

6.1.2.8 static uint16_t qsense::ByteOrder::fromBigEndian (uint16_t *value*) [static]

6.1.2.9 static int32_t qsense::ByteOrder::fromBigEndian (int32_t *value*) [static]

6.1.2.10 static uint32_t qsense::ByteOrder::fromBigEndian (uint32_t *value*) [static]

6.1.2.11 static int64_t qsense::ByteOrder::fromBigEndian (int64_t *value*) [static]

6.1.2.12 static uint64_t qsense::ByteOrder::fromBigEndian (uint64_t *value*) [static]

6.1.2.13 static int16_t qsense::ByteOrder::fromLittleEndian (int16_t *value*) [static]

6.1.2.14 static uint16_t qsense::ByteOrder::fromLittleEndian (uint16_t *value*) [static]

6.1.2.15 static int32_t qsense::ByteOrder::fromLittleEndian (int32_t *value*) [static]

6.1.2.16 static uint32_t qsense::ByteOrder::fromLittleEndian (uint32_t *value*) [static]

6.1.2.17 static int64_t qsense::ByteOrder::fromLittleEndian (int64_t *value*) [static]

6.1.2.18 static uint64_t qsense::ByteOrder::fromLittleEndian (uint64_t *value*) [static]

6.1.2.19 static int16_t qsense::ByteOrder::fromNetwork (int16_t *value*) [static]

6.1.2.20 static uint16_t qsense::ByteOrder::fromNetwork (uint16_t *value*) [static]

6.1.2.21 static int32_t qsense::ByteOrder::fromNetwork (int32_t *value*) [static]

6.1.2.22 static uint32_t qsense::ByteOrder::fromNetwork (uint32_t *value*) [static]

6.1.2.23 static int64_t qsense::ByteOrder::fromNetwork (int64_t *value*) [static]

6.1.2.24 static uint64_t qsense::ByteOrder::fromNetwork (uint64_t *value*) [static]

6.1.2.25 static int16_t qsense::ByteOrder::toBigEndian (int16_t *value*) [static]

6.1.2.26 static uint16_t qsense::ByteOrder::toBigEndian (uint16_t *value*) [static]

6.1.2.27 static int32_t qsense::ByteOrder::toBigEndian (int32_t *value*) [static]

6.1.2.28 static uint32_t qsense::ByteOrder::toBigEndian (uint32_t *value*) [static]

6.1.2.29 static int64_t qsense::ByteOrder::toBigEndian (int64_t *value*) [static]

6.1.2.30 static uint64_t qsense::ByteOrder::toBigEndian (uint64_t *value*) [static]

6.1.2.31 static int16_t qsense::ByteOrder::toLittleEndian (int16_t *value*) [static]

6.1.2.32 static uint16_t qsense::ByteOrder::toLittleEndian (uint16_t *value*) [static]

6.1.2.33 static int32_t qsense::ByteOrder::toLittleEndian (int32_t *value*) [static]

6.1.2.34 static uint32_t qsense::ByteOrder::toLittleEndian (uint32_t *value*) [static]

6.1.2.35 static int64_t qsense::ByteOrder::toLittleEndian (int64_t *value*) [static]

6.1.2.36 static uint64_t qsense::ByteOrder::toLittleEndian (uint64_t *value*) [static]

6.1.2.37 static int16_t qsense::ByteOrder::toNetwork (int16_t *value*) [static]

6.1.2.38 static uint16_t qsense::ByteOrder::toNetwork (uint16_t *value*) [static]

6.1.2.39 static int32_t qsense::ByteOrder::toNetwork (int32_t *value*) [static]

6.1.2.40 static uint32_t qsense::ByteOrder::toNetwork (uint32_t *value*) [static]

6.1.2.41 static int64_t qsense::ByteOrder::toNetwork (int64_t *value*) [static]

6.1.2.42 static uint64_t qsense::ByteOrder::toNetwork (uint64_t *value*) [static]

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

- /Users/rakesh/svn/customer/qsense/desktop/src/api/ByteOrder.h

6.2 qsense::hash::Sha1::Context Struct Reference

SHA1 context representation.

`#include <Sha1.h>`

Public Attributes

- unsigned long total [2]
- unsigned long state [5]
- unsigned char buffer [64]
- unsigned char ipad [64]
- unsigned char opad [64]

6.2.1 Detailed Description

SHA1 context representation.

6.2.2 Member Data Documentation

6.2.2.1 unsigned char qsense::hash::Sha1::Context::buffer[64]

data block being processed

6.2.2.2 unsigned char qsense::hash::Sha1::Context::ipad[64]

HMAC: inner padding

6.2.2.3 unsigned char qsense::hash::Sha1::Context::opad[64]

HMAC: outer padding

6.2.2.4 unsigned long qsense::hash::Sha1::Context::state[5]

intermediate digest state

6.2.2.5 unsigned long qsense::hash::Sha1::Context::total[2]

number of bytes processed

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

- /Users/rakesh/svn/customer/qsense/desktop/src/api/hash/Sha1.h

6.3 qsense::net::DateTime Class Reference

Represents current date/time. Seeds initially (and daily) from a network time service, and uses internal timer to represent a real-time clock.

`#include <DateTime.h>`

Public Member Functions

- DateTime ()

  Default constructor. Use singleton in general.
- const qsense::QString currentTime ()

  Return the current date/time in ISO 8601 format.
- const qsense::QString date ()

  Return the current date in ISO 8601 format.
- int64_t currentTimeMillis ()

  Return the milli seconds since UNIX epoch.

Static Public Member Functions

- static DateTime & singleton ()

  Return a singleton instance to use. This is the preferred way of using this class.

6.3.1 Detailed Description

Represents current date/time. Seeds initially (and daily) from a network time service, and uses internal timer to represent a real-time clock.

6.3.2 Constructor & Destructor Documentation

6.3.2.1 qsense::net::DateTime::DateTime ( )

Default constructor. Use singleton in general.

6.3.3 Member Function Documentation

6.3.3.1 const qsense::QString qsense::net::DateTime::currentTime ( )

Return the current date/time in ISO 8601 format.

6.3.3.2 int64_t qsense::net::DateTime::currentTimeMillis ( )

Return the milli seconds since UNIX epoch.

6.3.3.3 const qsense::QString qsense::net::DateTime::date ( )

Return the current date in ISO 8601 format.

6.3.3.4 static DateTime& qsense::net::DateTime::singleton ( ) [inline], [static]

Return a singleton instance to use. This is the preferred way of using this class.

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

- /Users/rakesh/svn/customer/qsense/desktop/src/api/net/DateTime.h

6.4 qsense::Event Class Reference

A simple class that encapsulates an event sent to Sidecar. Events are holders for readings. Events can be serialised to JSON using the toString method.

`#include <Event.h>`

Public Types

- typedef std::vector< qsense::Reading > Readings
  The vector of readings encapsulated in this event.
- typedef std::vector< qsense::QString > Tags
  The vector of tags associated with this event.
- typedef Readings::const_iterator ReadingsIterator
  Iterator for the readings encapsulated in this event.
- typedef Tags::const_iterator TagsIterator
  Iterator for the tags associated with this event.

Public Member Functions

- Event ()
  Default constructor. Uses default location set through init.
- Event (const qsense::Location &location)
  Create a new event with the specified location.
- ~Event ()
  Destructor. No actions required.
- Event & add (const qsense::Reading &reading)
  Add the specified reading to this event.
- Event & add (const qsense::QString &tag)
  Add the specified tag to this event.
- Event & operator+= (const qsense::Reading &reading)
  Operator for adding a reading to the event.
- Event & operator+= (const qsense::QString &tag)
  Operator for adding a tag to the event.
- std::size_t numberOfReadings () const
  Return the number of readings in this event.
- std::size_t numberOfTags () const
  Return the number of tags associated with this event.
- ReadingsIterator beginReadings () const
  Return a constant iterator to the beginning of the readings vector.
- ReadingsIterator endReadings () const
  The end of the readings vector to check in loops.
- TagsIterator beginTags () const
  Return a constant iterator to the beginning of the tags vector.
- TagsIterator endTags () const
  The end of the tags vector to check in loops.
- const qsense::Reading & operator[ ] (std::size_t index) const
  operator [] Retrieve the reading at specified index. Will throw exception if index is out of bounds. Check the size of the container before using this operator.
- const qsense::Location & getLocation () const
  Return the location used by this event.
- const qsense::QString toString () const
  Serialise the event to JSON.

Static Public Member Functions

- static void init (const qsense::QString &deviceId, const qsense::QString &stream, const qsense::Location &location)

  Initialise the Event API.

6.4.1 Detailed Description

A simple class that encapsulates an event sent to Sidecar. Events are holders for readings. Events can be serialised to JSON using the toString method.

6.4.2 Member Typedef Documentation

6.4.2.1 typedef std::vector<qsense::Reading> qsense::Event::Readings

The vector of readings encapsulated in this event.

6.4.2.2 typedef Readings::const_iterator qsense::Event::ReadingsIterator

Iterator for the readings encapsulated in this event.

6.4.2.3 typedef std::vector<qsense::QString> qsense::Event::Tags

The vector of tags associated with this event.

6.4.2.4 typedef Tags::const_iterator qsense::Event::TagsIterator

Iterator for the tags associated with this event.

6.4.3 Constructor & Destructor Documentation

6.4.3.1 qsense::Event::Event ( )

Default constructor. Uses default location set through init.

6.4.3.2 qsense::Event::Event (const qsense::Location & *location*)

Create a new event with the specified location.

6.4.3.3 qsense::Event::~Event ( ) [inline]

Destructor. No actions required.

6.4.4 Member Function Documentation

6.4.4.1 Event& qsense::Event::add (const qsense::Reading & *reading*)

Add the specified reading to this event.

6.4.4.2 Event& qsense::Event::add (const qsense::QString & *tag*)

Add the specified tag to this event.

6.4.4.3 ReadingsIterator qsense::Event::beginReadings ( ) const [inline]

Return a constant iterator to the beginning of the readings vector.

6.4.4.4 TagsIterator qsense::Event::beginTags ( ) const [inline]

Return a constant iterator to the beginning of the tags vector.

6.4.4.5 ReadingsIterator qsense::Event::endReadings ( ) const [inline]

The end of the readings vector to check in loops.

6.4.4.6 TagsIterator qsense::Event::endTags ( ) const [inline]

The end of the tags vector to check in loops.

6.4.4.7 const qsense::Location& qsense::Event::getLocation ( ) const [inline]

Return the location used by this event.

6.4.4.8 static void qsense::Event::init (const qsense::QString & *deviceId,* const qsense::QString & *stream,* const qsense::Location & *location*) [static]

Initialise the Event API.

Parameters

deviceId	The deviceId to use. No way at present to retrieve using API
stream	The stream identifier to use with Sidecar
location	A default location to use. No location tracking at present

6.4.4.9 std::size_t qsense::Event::numberOfReadings ( ) const [inline]

Return the number of readings in this event.

6.4.4.10 std::size_t qsense::Event::numberOfTags ( ) const [inline]

Return the number of tags associated with this event.

6.4.4.11 Event& qsense::Event::operator+= (const qsense::Reading & *reading*) [inline]

Operator for adding a reading to the event.

6.4.4.12 Event& qsense::Event::operator+= (const qsense::QString & *tag*) [inline]

Operator for adding a tag to the event.

6.4.4.13 const qsense::Reading& qsense::Event::operator[ ] (std::size_t *index*) const [inline]

operator [] Retrieve the reading at specified index. Will throw exception if index is out of bounds. Check the size of the container before using this operator.

Parameters

index	The index into the vector of readings.

Returns

The reading at the specified index.

6.4.4.14 const qsense::QString qsense::Event::toString ( ) const

Serialise the event to JSON.

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

- /Users/rakesh/svn/customer/qsense/desktop/src/api/Event.h

6.5 qsense::net::HttpClient Class Reference

A HTTP Client class for use with either ethernet or wifi. Before use, the client should be initialised with the network type used by the device (qsense::net::initNetworkType.

`#include <QHttpClient.h>`

Public Types

- typedef std::map< qsense::QString, qsense::QString > Headers

  Map used to define request headers.

Public Member Functions

- virtual ~HttpClient ()

  Destructor for sub-classes.
- virtual int16_t connect (const qsense::QString &server, uint16_t port=80)=0

  Make a socket connection to the specified server on specified port (default 80)
- virtual uint8_t connected ()=0

  Check to see if the client is connected to the server.
- virtual uint16_t get (const qsense::QString &uri, const Headers &headers=Headers())=0

  Perform a GET request for the specified url. Optionally specify a map of custom headers to send to server.
- virtual uint16_t post (const qsense::QString &uri, const Headers &headers=Headers(), const qsense::QString &body=qsense::QString())=0

  Perform a POST request to the specified url. Optionally specify a map of custom headers and a string body to send to server.
- virtual const qsense::QString readLine ()=0

  Read a line from the HTTP response.
- virtual Headers readHeaders ()=0

  Return a map of the HTTP response headers.
- virtual const qsense::QString readBody ()=0

  Read the entire contents of the server response body. Note: This method also reads headers. If headers have already been read, it may end up losing some of the response body.

Static Public Member Functions

- static HttpClient * create ()

  Factory method for creating concrete instances based on initialisation.

Protected Member Functions

- virtual void writeHeaders (const Headers &headers, bool close=true)=0

  Send the specified request headers to the HTTP server.

6.5.1 Detailed Description

A HTTP Client class for use with either ethernet or wifi. Before use, the client should be initialised with the network type used by the device (qsense::net::initNetworkType.

6.5.2 Member Typedef Documentation

6.5.2.1 typedef std::map<qsense::QString,qsense::QString> qsense::net::HttpClient::Headers

Map used to define request headers.

6.5.3 Constructor & Destructor Documentation

6.5.3.1 virtual qsense::net::HttpClient::~HttpClient ( ) [inline], [virtual]

Destructor for sub-classes.

6.5.4 Member Function Documentation

6.5.4.1 virtual int16_t qsense::net::HttpClient::connect (const qsense::QString & *server*, uint16_t *port* = 80) [pure virtual]

Make a socket connection to the specified server on specified port (default 80)

6.5.4.2 virtual uint8_t qsense::net::HttpClient::connected ( ) [pure virtual]

Check to see if the client is connected to the server.

6.5.4.3 static HttpClient* qsense::net::HttpClient::create ( ) [static]

Factory method for creating concrete instances based on initialisation.

Note: Callers must delete the returned instance.

Returns

An instance that uses either ethernet or wifi to connect to the network. Callers must delete the returned instance.

6.5.4.4 virtual uint16_t qsense::net::HttpClient::get (const qsense::QString & *uri,* const Headers & *headers =* Headers()) [pure virtual]

Perform a GET request for the specified url. Optionally specify a map of custom headers to send to server.

Parameters

uri	The remote URI path to request from server
headers	Optional map of request headers to specify to server.

Returns

The HTTP response code from server.

6.5.4.5 virtual uint16_t qsense::net::HttpClient::post (const qsense::QString & *uri*, const Headers & *headers* = Headers (), const qsense::QString & *body* = qsense::QString ()) [pure virtual]

Perform a POST request to the specified url. Optionally specify a map of custom headers and a string body to send to server.

Parameters

uri	The remote URI path to request from server
headers	Optional map of request headers to specify to server.
body	Optional body to post to server.

Returns

The HTTP response code from server.

6.5.4.6 virtual const qsense::QString qsense::net::HttpClient::readBody ( ) [pure virtual]

Read the entire contents of the server response body. Note: This method also reads headers. If headers have already been read, it may end up losing some of the response body.

WARNING: Use with caution. Can run embedded devices out of memory very easily.

Returns

The entire http response body content.

6.5.4.7 virtual Headers qsense::net::HttpClient::readHeaders ( ) [pure virtual]

Return a map of the HTTP response headers.

6.5.4.8 virtual const qsense::QString qsense::net::HttpClient::readLine ( ) [pure virtual]

Read a line from the HTTP response.

A line can be either a header or content. Use to process raw HTTP response line by line.

Returns

A line (content until newline character) of text from raw response.

6.5.4.9 virtual void qsense::net::HttpClient::writeHeaders (const Headers & *headers*, bool *close* = true) [protected], [pure virtual]

Send the specified request headers to the HTTP server.

Parameters

headers	The map of headers to send to the server.
close	Flag indicating whether HTTP keep-alive is NOT to be used.

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

- /Users/rakesh/svn/customer/qsense/desktop/src/api/net/QHttpClient.h

6.6 qsense::Location Class Reference

A simple representation of geographical location.

`#include <Location.h>`

Public Member Functions

- Location ()

  Default constructor.
- Location (float lat, float lon)

  Create a new instance with the specified co-ordinates.
- Location (const Location &location)

  Copy constructor.
- ~Location ()

  Destructor. No action required.
- Location & operator= (const Location &location)

  Copy assignment operator.
- const qsense::QString toString () const

  Serialise this instance to a JSON representation.
- float getLatitude () const
- float getLongitude () const

6.6.1 Detailed Description

A simple representation of geographical location.

6.6.2 Constructor & Destructor Documentation

6.6.2.1 qsense::Location::Location ( ) [inline]

Default constructor.

6.6.2.2 qsense::Location::Location (float *lat*, float *lon*) [inline]

Create a new instance with the specified co-ordinates.

Parameters

lat	The latitude

lon	The longitude

6.6.2.3 qsense::Location::Location (const Location & *location*) [inline]

Copy constructor.

6.6.2.4 qsense::Location::~Location ( ) [inline]

Destructor. No action required.

6.6.3 Member Function Documentation

6.6.3.1 float qsense::Location::getLatitude ( ) const [inline]

Returns

Return the latitude value

6.6.3.2 float qsense::Location::getLongitude ( ) const [inline]

Returns

Return the longitude value

6.6.3.3 Location& qsense::Location::operator= (const Location & *location*)

Copy assignment operator.

6.6.3.4 const qsense::QString qsense::Location::toString ( ) const

Serialise this instance to a JSON representation.

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

- /Users/rakesh/svn/customer/qsense/desktop/src/api/Location.h

6.7 qsense::hash::MD5 Class Reference

Class for generating MD5 hashes.

`#include <MD5.h>`

Public Types

- typedef qsense::Byte Byte

  8-bit byte
- typedef uint32_t Word

  32-bit word

Public Member Functions

- MD5 ()
  Default constructor.
- ~MD5 ()
  Destructor. Destroys the context.
- void compute (const Byte *data, Word nbytes, Byte digest[MD5_HASH_LENGTH])
- qsense::QString compute (const qsense::QString &input)

6.7.1 Detailed Description

Class for generating MD5 hashes.

6.7.2 Member Typedef Documentation

6.7.2.1 typedef qsense::Byte qsense::hash::MD5::Byte

8-bit byte

6.7.2.2 typedef uint32_t qsense::hash::MD5::Word

32-bit word

6.7.3 Constructor & Destructor Documentation

6.7.3.1 qsense::hash::MD5::MD5 ( ) [inline]

Default constructor.

6.7.3.2 qsense::hash::MD5::~MD5 ( ) [inline]

Destructor. Destroys the context.

6.7.4 Member Function Documentation

6.7.4.1 void qsense::hash::MD5::compute (const Byte * *data*, Word *nbytes*, Byte *digest[MD5_HASH_LENGTH]*)

Compute the MD5 digest for the specified data of length nbytes into digest

6.7.4.2 qsense::QString qsense::hash::MD5::compute (const qsense::QString & *input*)

Compute MD5 digest for specified data and return base64 encoded string

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

- /Users/rakesh/svn/customer/qsense/desktop/src/api/hash/MD5.h

6.8 qsense::Reading Class Reference

A class that represents a single reading. Readings are added to an Event.

`#include <Reading.h>`

Public Member Functions

- Reading (const qsense::QString &k, const qsense::QString &v, const qsense::QString &ts=qsense::net::↩ DateTime::singleton().currentTime())

  Create a new reading with specified values.
- Reading (const qsense::QString &k, float v, const qsense::QString &ts=qsense::net::DateTime↩ ::singleton().currentTime())

  Create a new reading with specified values.
- ~Reading ()

  Destructor. No actions required.
- const qsense::QString & getKey () const

  Return the key for the reading.
- const qsense::QString & getValue () const

  Return the value of the reading.
- const qsense::QString & getTimestamp () const

  Return the time at which the reading was taken.
- const qsense::QString toString () const

  Return a JSON representation of the reading.

6.8.1 Detailed Description

A class that represents a single reading. Readings are added to an Event.

6.8.2 Constructor & Destructor Documentation

6.8.2.1 qsense::Reading::Reading (const qsense::QString & *k*, const qsense::QString & *v*, const qsense::QString & *ts* = qsense::net::DateTime::singleton().currentTime()) [inline]

Create a new reading with specified values.

Parameters

k	The key to associate with the reading
v	The value of the reading
ts	The timestamp (optional) at which reading was taken.

6.8.2.2 qsense::Reading::Reading (const qsense::QString & *k*, float *v*, const qsense::QString & *ts* = qsense::net::DateTime::singleton().currentTime())

Create a new reading with specified values.

Parameters

k	The key to associate with the reading
v	The float value of the reading
ts	The timestamp (optional) at which reading was taken.

6.8.2.3 qsense::Reading::~Reading ( ) [inline]

Destructor. No actions required.

6.8.3 Member Function Documentation

6.8.3.1 const qsense::QString& qsense::Reading::getKey ( ) const [inline]

Return the key for the reading.

6.8.3.2 const qsense::QString& qsense::Reading::getTimestamp ( ) const [inline]

Return the time at which the reading was taken.

6.8.3.3 const qsense::QString& qsense::Reading::getValue ( ) const [inline]

Return the value of the reading.

6.8.3.4 const qsense::QString qsense::Reading::toString ( ) const

Return a JSON representation of the reading.

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

- /Users/rakesh/svn/customer/qsense/desktop/src/api/Reading.h

6.9 qsense::hash::Sha1 Class Reference

Class for hashing using SHA1 algorithm.

`#include <Sha1.h>`

Classes

- struct Context

  SHA1 context representation.

Public Member Functions

- Sha1 ()

  Default constructor.
- ~Sha1 ()

  Destructor. No actions required.
- void hash (unsigned char *input, int ilen, unsigned char output[20])

  Generate SHA1 hash for specified input into output array.
- qsense::QString hash (const qsense::QString &input)

  Generate SHA1 hash for the specified input string.
- void hmac (unsigned char *key, int keylen, unsigned char *input, int ilen, unsigned char output[20])

  Generate SHA1 HMAC for the specified input using specified key.
- qsense::QString hmac (const qsense::QString &key, const qsense::QString &input)

  Generate SHA1 HMAC using the specified key for the input string.
- qsense::QString sign (const qsense::QString &privateKey, const qsense::QString &httpMethod, const qsense::QString &uriPath, const qsense::QString &date, const qsense::QString &contentMd5, const qsense::QString &signatureVersion=qsense::QString("1"))

  Generate the signature for the Sidecar Authorization header.

6.9.1 Detailed Description

Class for hashing using SHA1 algorithm.

6.9.2 Constructor & Destructor Documentation

6.9.2.1 qsense::hash::Sha1::Sha1 ( )

Default constructor.

6.9.2.2 qsense::hash::Sha1::~Sha1 ( ) [inline]

Destructor. No actions required.

6.9.3 Member Function Documentation

6.9.3.1 void qsense::hash::Sha1::hash (unsigned char * *input*, int *ilen*, unsigned char *output[20]*)

Generate SHA1 hash for specified input into output array.

Parameters

input	The input char array that is to be hashed.
ilen	The length of the input char array.
output	The output char array into which hash value is written.

6.9.3.2 qsense::QString qsense::hash::Sha1::hash (const qsense::QString & *input*)

Generate SHA1 hash for the specified input string.

6.9.3.3 void qsense::hash::Sha1::hmac (unsigned char * *key*, int *keylen*, unsigned char * *input*, int *ilen*, unsigned char *output[20]*)

Generate SHA1 HMAC for the specified input using specified key.

Parameters

key	The key to use to generate the HMAC
keylen	The length of the key
input	The input char array to hash
ilen	The length of the input char array
output	The output char array into which hash value is written.

6.9.3.4 qsense::QString qsense::hash::Sha1::hmac (const qsense::QString & *key*, const qsense::QString & *input*)

Generate SHA1 HMAC using the specified key for the input string.

6.9.3.5 qsense::QString qsense::hash::Sha1::sign (const qsense::QString & *privateKey*, const qsense::QString & *httpMethod*, const qsense::QString & *uriPath*, const qsense::QString & *date*, const qsense::QString & *contentMd5*, const qsense::QString & *signatureVersion* = qsense::QString ("1"))

Generate the signature for the Sidecar Authorization header.

Parameters

privateKey	The api secret to use to sign
httpMethod	The HTTP method used for the Sidecar API interaction
uriPath	The URI path with which to interact
date	The current timestamp
contentMd5	The MD5 hash for the content to be submitted to Sidecar
signatureVersion	The signature version to specify in header

Returns

The Base64 encoded authorisation signature

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

- /Users/rakesh/svn/customer/qsense/desktop/src/api/hash/Sha1.h

6.10 qsense::net::SidecarClient Class Reference

Class that encapsulates interactions with the Sidecar REST API.

`#include <SidecarClient.h>`

Public Member Functions

- bool publish (const qsense::Event &event) const

  Publish the specified event to the Sidecar Event API.

Static Public Member Functions

- static void init (const qsense::QString &apiKey, const qsense::QString &apiSecret)

  Initialise the API with the key and secret used to sign messages.

6.10.1 Detailed Description

Class that encapsulates interactions with the Sidecar REST API.

6.10.2 Member Function Documentation

6.10.2.1 static void qsense::net::SidecarClient::init (const qsense::QString & *apiKey*, const qsense::QString & *apiSecret*) [static]

Initialise the API with the key and secret used to sign messages.

6.10.2.2 bool qsense::net::SidecarClient::publish (const qsense::Event & *event*) const

Publish the specified event to the Sidecar Event API.

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

- /Users/rakesh/svn/customer/qsense/desktop/src/api/net/SidecarClient.h

6.11 qsense::UUID Class Reference

A class that represents a UUID/GUID.

`#include <UUID.h>`

Public Types

- enum Version { UUID_TIME_BASED = 0x01, UUID_DCE_UID = 0x02, UUID_NAME_BASED = 0x03, UUID_RANDOM = 0x04 }

Public Member Functions

- UUID ()

  Creates a nil (all zero) UUID.
- UUID (const UUID &uuid)

  Copy constructor.
- UUID (const QString &uuid)

  Parses the UUID from a string.
- UUID (const char *uuid)

  Parses the UUID from a char array.
- ~UUID ()

  Destroys the UUID.
- UUID & operator= (const UUID &uuid)

  Assignment operator.
- bool parse (const QString &uuid)

  Tries to interpret the given string as an UUID.
- QString toString () const

  Returns a string representation of the UUID consisting of groups of hexadecimal digits separated by hyphens.
- void copyFrom (const char *buffer)

  Copies the UUID (16 bytes) from a buffer or byte array. The UUID fields are expected to be stored in network byte order.
- void copyTo (char *buffer) const

  Copies the UUID to the buffer. The fields are in network byte order. The buffer need not be aligned.
- Version version () const

  Returns the version of the UUID.
- int variant () const

  Returns the variant number of the UUID:
- bool operator== (const UUID &uuid) const
- bool operator!= (const UUID &uuid) const
- bool operator< (const UUID &uuid) const
- bool operator<= (const UUID &uuid) const
- bool operator> (const UUID &uuid) const
- bool operator>= (const UUID &uuid) const
- bool isNull () const

Static Public Member Functions

- static const UUID & null ()

  Returns a null/nil UUID.
- static const UUID & dns ()

  Returns the namespace identifier for the DNS namespace.
- static const UUID & uri ()

  Returns the namespace identifier for the URI (former URL) namespace.
- static const UUID & oid ()

  Returns the namespace identifier for the OID namespace.
- static const UUID & x500 ()

  Returns the namespace identifier for the X500 namespace.
- static const UUID create ()

  Generate a time based UUID instance.
- static void init (uint8_t node[6])

  Initialise the UUID engine. On application start, invoke with the current MAC address.

Protected Member Functions

- UUID (uint32_t timeLow, uint32_t timeMid, uint32_t timeHiAndVersion, uint16_t clockSeq, uint8_t node[6])
- UUID (const char *bytes, Version version)
- int compare (const UUID &uuid) const
- void fromNetwork ()
- void toNetwork ()

Static Protected Member Functions

- static void appendHex (QString &str, uint8_t n)
- static void appendHex (QString &str, uint16_t n)
- static void appendHex (QString &str, uint32_t n)
- static uint8_t nibble (char hex)
- static uint32_t randomNumber (int32_t input)

6.11.1 Detailed Description

A class that represents a UUID/GUID.

A UUID is an identifier that is unique across both space and time, with respect to the space of all UUIDs. Since a UUID is a fixed size and contains a time field, it is possible for values to rollover (around A.D. 3400, depending on the specific algorithm used). A UUID can be used for multiple purposes, from tagging objects with an extremely short lifetime, to reliably identifying very persistent objects across a network.

This class implements a Universal Unique Identifier, as specified in Appendix A of the DCE 1.1 Remote Procedure Call Specification (http://www.opengroup.org/onlinepubs/9629399/), RFC 2518 (WebDAV), section 6.4.1 and the UUIDs and GUIDs internet draft by Leach/Salz from February, 1998 (http://www.ics.uci.edu/~ejw/authoring/uuid-guid/draft-leach-uuids-guids-01.txt) and also http://tools.ietf.org/html/draft-mealling-uuid-urn-05

6.11.2 Member Enumeration Documentation

6.11.2.1 enum qsense::UUID::Version

Enumerator

UUID_TIME_BASED

UUID_DCE_UID

UUID_NAME_BASED

UUID_RANDOM

6.11.3 Constructor & Destructor Documentation

6.11.3.1 qsense::UUID::UUID ( )

Creates a nil (all zero) UUID.

6.11.3.2 qsense::UUID::UUID (const UUID & *uuid*)

Copy constructor.

6.11.3.3 qsense::UUID::UUID (const QString & *uuid*) [explicit]

Parses the UUID from a string.

6.11.3.4 qsense::UUID::UUID (const char * *uuid*) [explicit]

Parses the UUID from a char array.

6.11.3.5 qsense::UUID::~UUID ( )

Destroys the UUID.

6.11.3.6 qsense::UUID::UUID (uint32_t *timeLow*, uint32_t *timeMid*, uint32_t *timeHiAndVersion*, uint16_t *clockSeq*, uint8_t *node[6]*) [protected]

6.11.3.7 qsense::UUID::UUID (const char * *bytes*, Version *version*) [protected]

6.11.4 Member Function Documentation

6.11.4.1 static void qsense::UUID::appendHex (QString & *str*, uint8_t *n*) [static], [protected]

6.11.4.2 static void qsense::UUID::appendHex (QString & *str*, uint16_t *n*) [static], [protected]

6.11.4.3 static void qsense::UUID::appendHex (QString & *str*, uint32_t *n*) [static], [protected]

6.11.4.4 int qsense::UUID::compare (const UUID & *uuid*) const [protected]

6.11.4.5 void qsense::UUID::copyFrom (const char * *buffer*)

Copies the UUID (16 bytes) from a buffer or byte array. The UUID fields are expected to be stored in network byte order.

Parameters

buffer	The buffer need not be aligned.

6.11.4.6 void qsense::UUID::copyTo (char * *buffer*) const

Copies the UUID to the buffer. The fields are in network byte order. The buffer need not be aligned.

Parameters

buffer	There must be room for at least 16 bytes.

6.11.4.7 static const UUID qsense::UUID::create ( ) [static]

Generate a time based UUID instance.

6.11.4.8 static const UUID& qsense::UUID::dns ( ) [static]

Returns the namespace identifier for the DNS namespace.

6.11.4.9 void qsense::UUID::fromNetwork ( ) [protected]

6.11.4.10 static void qsense::UUID::init (uint8_t *node[6]*) [static]

Initialise the UUID engine. On application start, invoke with the current MAC address.

Parameters

node	The MAC address.

6.11.4.11 bool qsense::UUID::isNull ( ) const [inline]

Returns

Returns true if the UUID is nil (in other words, consists of all zeros).

6.11.4.12 static uint8_t qsense::UUID::nibble (char *hex*) [static], [protected]

6.11.4.13 static const UUID& qsense::UUID::null ( ) [static]

Returns a null/nil UUID.

6.11.4.14 static const UUID& qsense::UUID::oid ( ) [static]

Returns the namespace identifier for the OID namespace.

6.11.4.15 bool qsense::UUID::operator!= (const UUID & *uuid*) const [inline]

6.11.4.16 bool qsense::UUID::operator< (const UUID & *uuid*) const [inline]

6.11.4.17 bool qsense::UUID::operator<= (const UUID & *uuid*) const [inline]

6.11.4.18 UUID& qsense::UUID::operator= (const UUID & *uuid*)

Assignment operator.

6.11.4.19 bool qsense::UUID::operator== (const UUID & *uuid*) const [inline]

6.11.4.20 bool qsense::UUID::operator> (const UUID & *uuid*) const [inline]

6.11.4.21 bool qsense::UUID::operator>= (const UUID & *uuid*) const [inline]

6.11.4.22 bool qsense::UUID::parse (const QString & *uuid*)

Tries to interpret the given string as an UUID.

Parameters

uuid	The value to parse

Returns

If the UUID is syntactically valid, assigns the members and returns true. Otherwise leaves the object unchanged and returns false.

6.11.4.23 static uint32_t qsense::UUID::randomNumber (int32_t *input*) [static], [protected]

6.11.4.24 void qsense::UUID::toNetwork ( ) [protected]

6.11.4.25 QString qsense::UUID::toString ( ) const

Returns a string representation of the UUID consisting of groups of hexadecimal digits separated by hyphens.

6.11.4.26 static const UUID& qsense::UUID::uri ( ) [static]

Returns the namespace identifier for the URI (former URL) namespace.

6.11.4.27 int qsense::UUID::variant ( ) const

Returns the variant number of the UUID:

Returns

- 0 reserved for NCS backward compatibility
- 2 the Leach-Salz variant (used by this class)
- 6 reserved, Microsoft Corporation backward compatibility
- 7 reserved for future definition

6.11.4.28 UUID::Version qsense::UUID::version ( ) const [inline]

Returns the version of the UUID.

6.11.4.29 static const UUID& qsense::UUID::x500 ( ) [static]

Returns the namespace identifier for the X500 namespace.

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

- /Users/rakesh/svn/customer/qsense/desktop/src/api/UUID.h

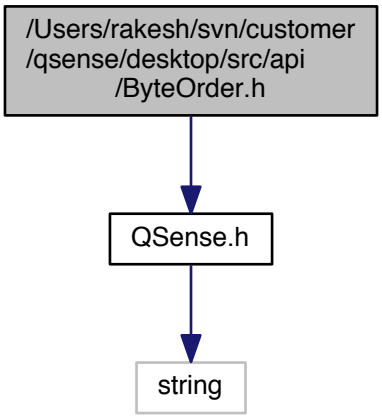
Chapter 7

File Documentation

7.1 /Users/rakesh/svn/customer/qsense/desktop/src/api/ByteOrder.h File Reference

```
#include <QSense.h>
```

Include dependency graph for ByteOrder.h:



Classes

- class qsense::ByteOrder

  This class contains a number of static methods to convert between big-endian and little-endian integers of various sizes.

Namespaces

- qsense

Macros

- #define IMPLEMENT_BYTEORDER_NOOP_(op, type)

- #define IMPLEMENT_BYTEORDER_FLIP_(op, type)
- #define IMPLEMENT_BYTEORDER_NOOP(op)
- #define IMPLEMENT_BYTEORDER_FLIP(op)
- #define IMPLEMENT_BYTEORDER_BIG IMPLEMENT_BYTEORDER_FLIP
- #define IMPLEMENT_BYTEORDER_LIT IMPLEMENT_BYTEORDER_NOOP

7.1.1 Macro Definition Documentation

7.1.1.1 #define IMPLEMENT_BYTEORDER_BIG IMPLEMENT_BYTEORDER_FLIP

7.1.1.2 #define IMPLEMENT_BYTEORDER_FLIP(*op*)

Value:

```
IMPLEMENT_BYTEORDER_FLIP_(op, int16_t) \
    IMPLEMENT_BYTEORDER_FLIP_(op, uint16_t)    \
    IMPLEMENT_BYTEORDER_FLIP_(op, int32_t) \
    IMPLEMENT_BYTEORDER_FLIP_(op, uint32_t)    \
    IMPLEMENT_BYTEORDER_FLIP_(op, int64_t) \
    IMPLEMENT_BYTEORDER_FLIP_(op, uint64_t)
```

7.1.1.3 #define IMPLEMENT_BYTEORDER_FLIP_(*op, type*)

Value:

```
inline type ByteOrder::op(type value)       \
    {                                           \
        return flipBytes(value);                \
    }
```

7.1.1.4 #define IMPLEMENT_BYTEORDER_LIT IMPLEMENT_BYTEORDER_NOOP

7.1.1.5 #define IMPLEMENT_BYTEORDER_NOOP(*op*)

Value:

```
IMPLEMENT_BYTEORDER_NOOP_(op, int16_t) \
    IMPLEMENT_BYTEORDER_NOOP_(op, uint16_t)    \
    IMPLEMENT_BYTEORDER_NOOP_(op, int32_t) \
    IMPLEMENT_BYTEORDER_NOOP_(op, uint32_t)    \
    IMPLEMENT_BYTEORDER_NOOP_(op, int64_t) \
    IMPLEMENT_BYTEORDER_NOOP_(op, uint64_t)
```

7.1.1.6 #define IMPLEMENT_BYTEORDER_NOOP_(*op, type*)

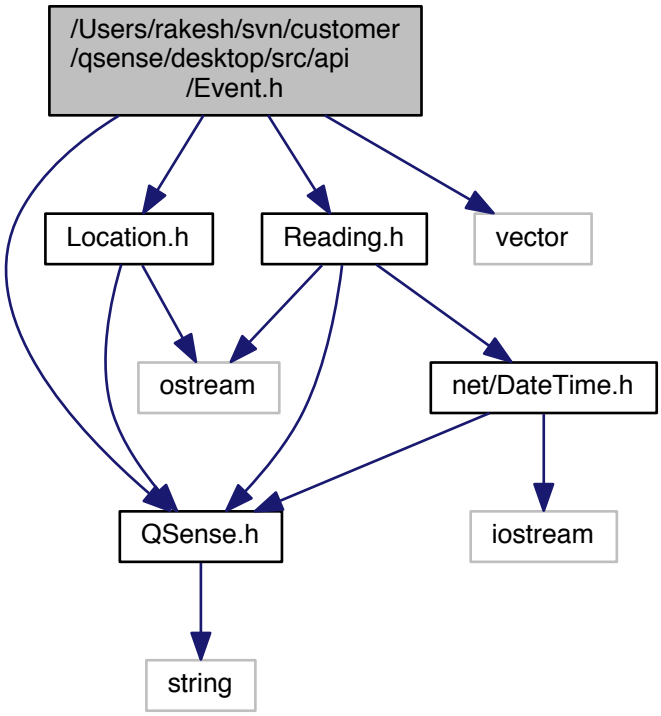
Value:

```
inline type ByteOrder::op(type value)       \
    {                                           \
        return value;                           \
    }
```

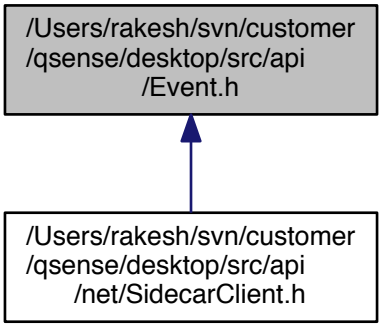
7.2 /Users/rakesh/svn/customer/qsense/desktop/src/api/Event.h File Reference

```
#include <QSense.h>
#include <Location.h>
#include <Reading.h>
#include <vector>
```

Include dependency graph for Event.h:



This graph shows which files directly or indirectly include this file:



Classes

- class qsense::Event

  A simple class that encapsulates an event sent to Sidecar. Events are holders for readings. Events can be serialised to JSON using the toString method.

Namespaces

- qsense

Functions

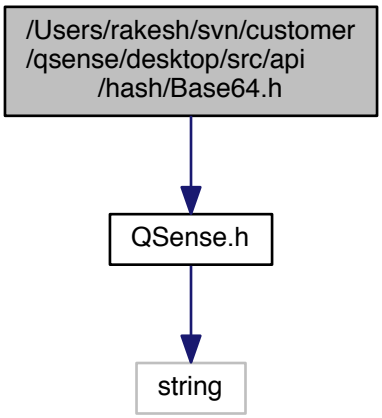
- std::ostream & qsense::operator<< (std::ostream &os, const Event &event)

  Serialise the specified event as JSON to the output stream.

7.3 /Users/rakesh/svn/customer/qsense/desktop/src/api/hash/Base64.h File Reference

```
#include <QSense.h>
```

Include dependency graph for Base64.h:



Namespaces

- qsense
- qsense::hash
- qsense::hash::base64

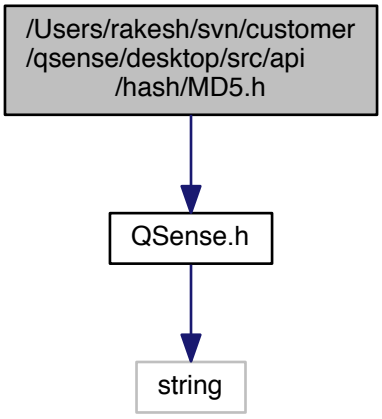
Functions

- int32_t qsense::hash::base64::encodeLength (int32_t len)
- int32_t qsense::hash::base64::encode (char ∗output, const char ∗input, int32_t inputLength)
- int32_t qsense::hash::base64::decodeLength (const char ∗code)
- int32_t qsense::hash::base64::decode (char ∗outputPlainText, const char ∗encoded)

7.4 /Users/rakesh/svn/customer/qsense/desktop/src/api/hash/MD5.h File Reference

```
#include <QSense.h>
```

Include dependency graph for MD5.h:



Classes

- class qsense::hash::MD5

  Class for generating MD5 hashes.

Namespaces

- qsense
- qsense::hash

Macros

- #define MD5_HASH_LENGTH 16

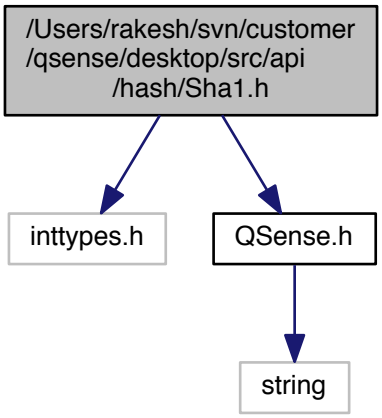
7.4.1 Macro Definition Documentation

7.4.1.1 #define MD5_HASH_LENGTH 16

7.5 /Users/rakesh/svn/customer/qsense/desktop/src/api/hash/Sha1.h File Reference

```
#include <inttypes.h>
#include <QSense.h>
```

Include dependency graph for Sha1.h:



Classes

- class qsense::hash::Sha1

  Class for hashing using SHA1 algorithm.

- struct qsense::hash::Sha1::Context

  SHA1 context representation.

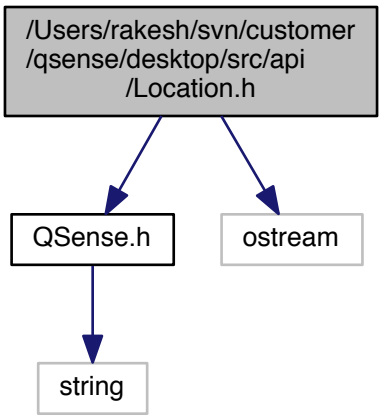
Namespaces

- qsense
- qsense::hash

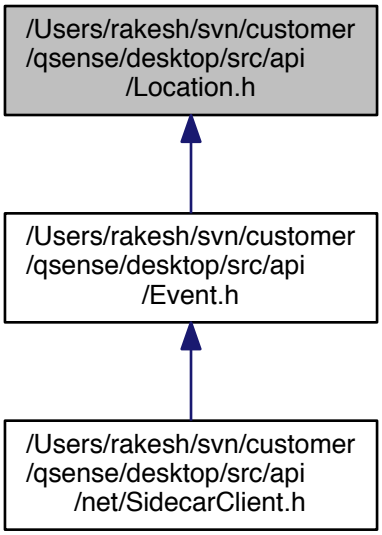
7.6 /Users/rakesh/svn/customer/qsense/desktop/src/api/Location.h File Reference

```
#include <QSense.h>
#include <ostream>
```

Include dependency graph for Location.h:



This graph shows which files directly or indirectly include this file:



Classes

- class qsense::Location

  A simple representation of geographical location.

Namespaces

- qsense

Functions

- std::ostream & qsense::operator<< (std::ostream &os, const qsense::Location &location)

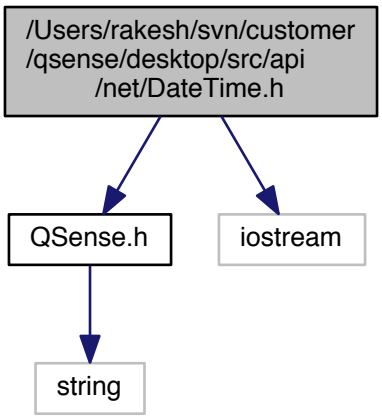
  Serialise the specified location to the output stream.

7.7 /Users/rakesh/svn/customer/qsense/desktop/src/api/mainpage.dox File Reference

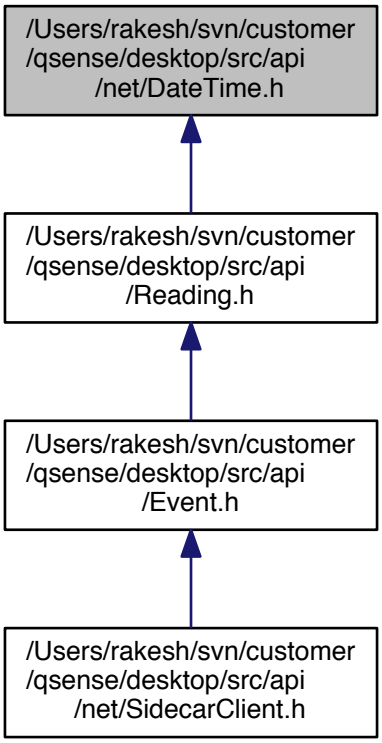
7.8 /Users/rakesh/svn/customer/qsense/desktop/src/api/net/DateTime.h File Reference

```
#include <QSense.h>
#include <iostream>
```

Include dependency graph for DateTime.h:



This graph shows which files directly or indirectly include this file:



Classes

- class qsense::net::DateTime

  Represents current date/time. Seeds initially (and daily) from a network time service, and uses internal timer to represent a real-time clock.

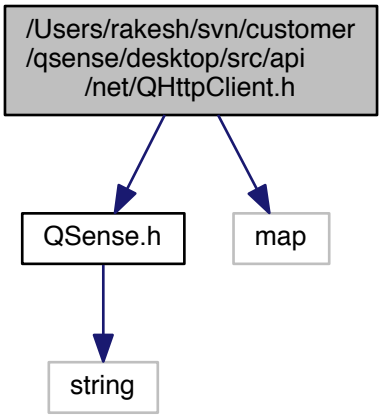
Namespaces

- qsense
- qsense::net

7.9 /Users/rakesh/svn/customer/qsense/desktop/src/api/net/QHttpClient.h File Reference

```
#include <QSense.h>
#include <map>
```

Include dependency graph for QHttpClient.h:



Classes

- class qsense::net::HttpClient

  A HTTP Client class for use with either ethernet or wifi. Before use, the client should be initialised with the network type used by the device (qsense::net::initNetworkType.

Namespaces

- qsense
- qsense::net

Enumerations

- enum qsense::net::NetworkType { qsense::net::Ethernet = 0, qsense::net::WiFi = 1 }

  Enumeration of network connection types for device.

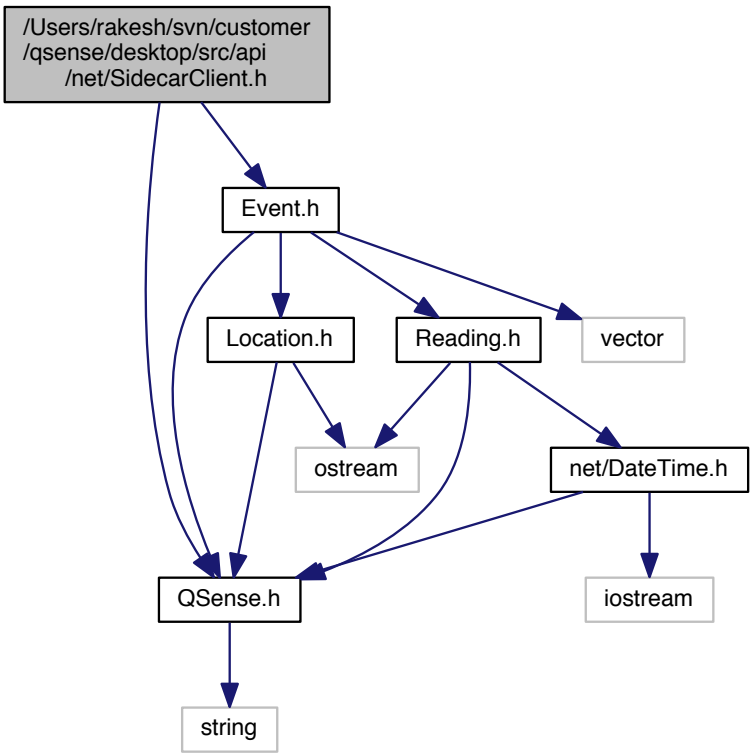
Functions

- void qsense::net::initNetworkType (NetworkType type)

7.10 /Users/rakesh/svn/customer/qsense/desktop/src/api/net/SidecarClient.h File Reference

```
#include <QSense.h>
#include <Event.h>
```

Include dependency graph for SidecarClient.h:



Classes

- class qsense::net::SidecarClient

  Class that encapsulates interactions with the Sidecar REST API.

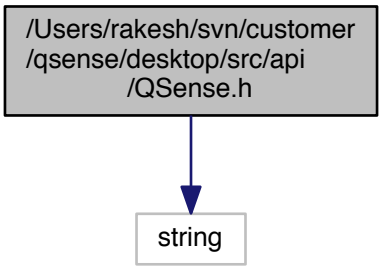
Namespaces

- qsense
- qsense::net

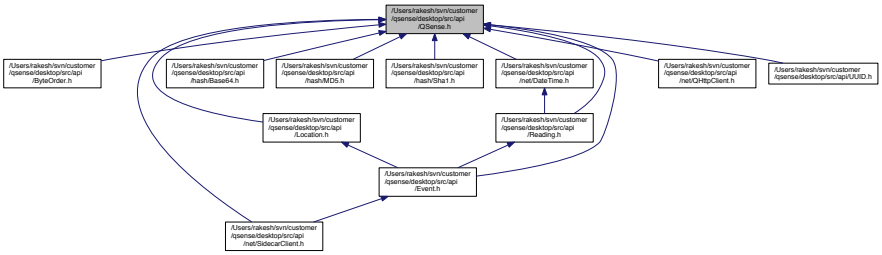
7.11 /Users/rakesh/svn/customer/qsense/desktop/src/api/QSense.h File Reference

```
#include <string>
```

Include dependency graph for QSense.h:



This graph shows which files directly or indirectly include this file:



Namespaces

- qsense

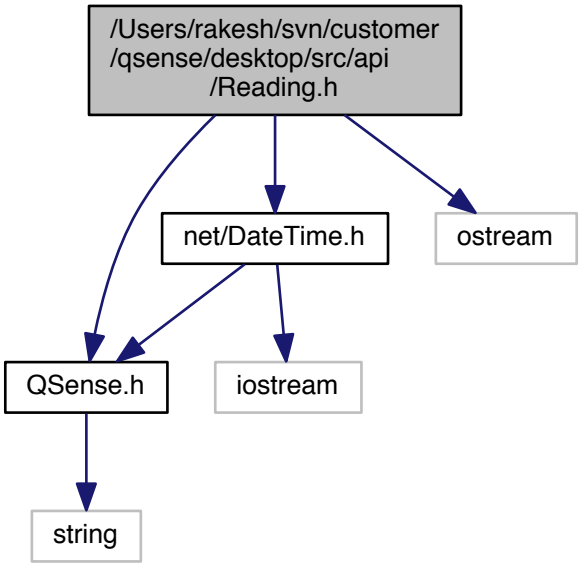
Typedefs

- typedef std::string qsense::QString
- typedef unsigned char qsense::Byte

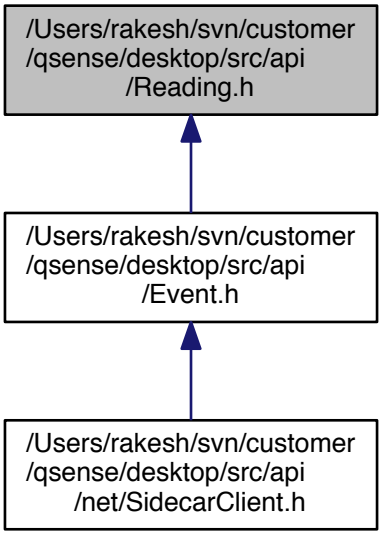
7.12 /Users/rakesh/svn/customer/qsense/desktop/src/api/Reading.h File Reference

```
#include <QSense.h>
#include <net/DateTime.h>
#include <ostream>
```

Include dependency graph for Reading.h:



This graph shows which files directly or indirectly include this file:



Classes

- class qsense::Reading

  A class that represents a single reading. Readings are added to an Event.

Namespaces

- qsense

Functions

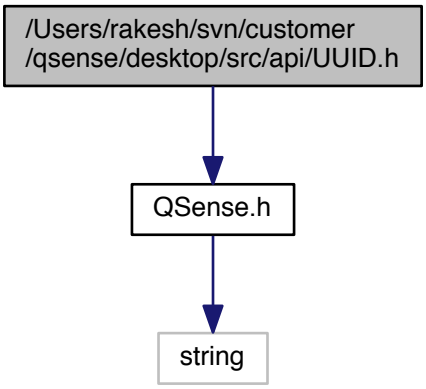
- std::ostream & qsense::operator<< (std::ostream &os, const qsense::Reading &reading)

  Serialise the reading as JSON to the output stream.

7.13 /Users/rakesh/svn/customer/qsense/desktop/src/api/UUID.h File Reference

```
#include <QSense.h>
```

Include dependency graph for UUID.h:



Classes

- class qsense::UUID

  A class that represents a UUID/GUID.

Namespaces

- qsense