

C++ UTC Time Library

Generated by Doxygen 1.8.1.2

Wed Aug 28 2013 14:03:14

Contents

1	Todo List	1
2	Namespace Index	3
2.1	Namespace List	3
3	Class Index	5
3.1	Class Hierarchy	5
4	Class Index	7
4.1	Class List	7
5	File Index	9
5.1	File List	9
6	Namespace Documentation	11
6.1	utctime Namespace Reference	11
6.1.1	Function Documentation	12
6.1.1.1	check_utc_timestamp	12
6.1.1.2	get_day_diff	13
6.1.1.3	get_hour_diff	13
6.1.1.4	get_sec_diff	13
6.1.1.5	get_utc_timestamp	13
6.1.1.6	get_utc_timestamp_sec_diff	14
6.1.1.7	is_leap_year	14
6.1.1.8	tm_compare	14
6.1.1.9	tm_decrement_day	15
6.1.1.10	tm_decrement_hour	15
6.1.1.11	tm_decrement_minute	15
6.1.1.12	tm_decrement_second	16
6.1.1.13	tm_increment_day	16
6.1.1.14	tm_increment_hour	16
6.1.1.15	tm_increment_minute	16
6.1.1.16	tm_increment_second	17

6.1.1.17	tm_intraday_secs_diff	17
6.1.1.18	validate_date	17
7	Class Documentation	19
7.1	utctime::bad_time Class Reference	19
7.2	utctime::bad_time_init Class Reference	20
7.3	utctime::invalid_date Class Reference	21
7.4	utctime::UTCTime Class Reference	21
7.4.1	Detailed Description	22
7.4.2	Constructor & Destructor Documentation	22
7.4.2.1	UTCTime	22
7.4.2.2	UTCTime	22
7.4.2.3	UTCTime	23
7.4.3	Member Function Documentation	23
7.4.3.1	get_tm	23
7.4.3.2	operator!=	23
7.4.3.3	operator-	23
7.4.3.4	operator<	24
7.4.3.5	operator<=	24
7.4.3.6	operator==	24
7.4.3.7	operator>	24
7.4.3.8	operator>=	25
7.4.3.9	time_string	25
7.4.3.10	time_string_inet	25
7.4.3.11	timestamp	25
7.5	utctime::UTCTimeException Class Reference	26
7.5.1	Detailed Description	26
7.5.2	Constructor & Destructor Documentation	26
7.5.2.1	UTCTimeException	26
7.5.3	Member Function Documentation	26
7.5.3.1	what	26
8	File Documentation	27
8.1	utctime.cpp File Reference	27
8.1.1	Detailed Description	27
8.2	utctime.h File Reference	28
8.2.1	Detailed Description	30

Chapter 1

Todo List

Member `utctime::validate_date` (const int year, const int month, const int day, const int hour, const int minute, const int second)

Why does this throw an exception? Why not return false? In order so that the error message can show why it's invalid?

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

utctime	Namespace for UTCTime() class and associated functions	11
-------------------------	--	--------------------

Chapter 3

Class Index

3.1 Class Hierarchy

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

utctime::UTCTime	21
utctime::UTCTimeException	26
utctime::bad_time	19
utctime::bad_time_init	20
utctime::invalid_date	21

Chapter 4

Class Index

4.1 Class List

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

utctime::bad_time	
Thrown when the time cannot be obtained	19
utctime::bad_time_init	
Thrown when the time cannot be initialized	20
utctime::invalid_date	
Thrown when an invalid date is provided to the constructor	21
utctime::UTCTime	21
utctime::UTCTimeException	
Base exception class	26

Chapter 5

File Index

5.1 File List

Here is a list of all documented files with brief descriptions:

utctime.cpp	Implementation of UTCTime() class and associated functions	27
utctime.h	Interface to UTCTime() class and associated functions	28

Chapter 6

Namespace Documentation

6.1 utctime Namespace Reference

Namespace for UTCTime() class and associated functions.

Classes

- class [UTCTimeException](#)
Base exception class.
- class [bad_time_init](#)
Thrown when the time cannot be initialized.
- class [bad_time](#)
Thrown when the time cannot be obtained.
- class [invalid_date](#)
Thrown when an invalid date is provided to the constructor.
- class [UTCTime](#)

Functions

- bool [validate_date](#) (const int year, const int month, const int day, const int hour, const int minute, const int second)
Checks whether a supplied date is valid.
- time_t [get_day_diff](#) ()
Returns a time_t interval representing one day.
- time_t [get_hour_diff](#) ()
Returns a time_t interval representing one hour.
- time_t [get_sec_diff](#) ()
Returns a time_t interval representing one second.
- int [tm_compare](#) (const std::tm &first, const std::tm &second)
Compares two std::tm structs.
- int [tm_intraday_secs_diff](#) (const std::tm &first, const std::tm &second)
Returns the difference between two std::tm structs.
- bool [is_leap_year](#) (const int year)
Checks if the supplied year is a leap year.
- std::tm * [tm_increment_day](#) (std::tm *changing_tm, const int quantity=1)
Adds one or more days to a std::tm struct.
- std::tm * [tm_increment_hour](#) (std::tm *changing_tm, const int quantity=1)

- Adds one or more hours to a std::tm struct.*
- std::tm * [tm_increment_minute](#) (std::tm *changing_tm, const int quantity=1)
- Adds one or more minutes to a std::tm struct.*
- std::tm * [tm_increment_second](#) (std::tm *changing_tm, const int quantity=1)
- Adds one or more seconds to a std::tm struct.*
- std::tm * [tm_decrement_day](#) (std::tm *changing_tm, const int quantity=1)
- Deducts one or more days from a std::tm struct.*
- std::tm * [tm_decrement_hour](#) (std::tm *changing_tm, const int quantity=1)
- Deducts one or more hours from a std::tm struct.*
- std::tm * [tm_decrement_minute](#) (std::tm *changing_tm, const int quantity=1)
- Deducts one or more minutes from a std::tm struct.*
- std::tm * [tm_decrement_second](#) (std::tm *changing_tm, const int quantity=1)
- Deducts one or more seconds from a std::tm struct.*
- bool [check_utc_timestamp](#) (const time_t check_time, int &secs_diff, const int year, const int month, const int day, const int hour, const int minute, const int second)
- Checks if a UTC timestamp is accurate.*
- time_t [get_utc_timestamp](#) (const int year, const int month, const int day, const int hour, const int minute, const int second)
- Gets a time_t timestamp for a requested UTC time.*
- int [get_utc_timestamp_sec_diff](#) (const time_t check_time, const int year, const int month, const int day, const int hour, const int minute, const int second)
- Checks a time_t timestamp against a UTC time.*

6.1.1 Function Documentation

6.1.1.1 bool `utctime::check_utc_timestamp` (const time_t *check_time*, int & *secs_diff*, const int *year*, const int *month*, const int *day*, const int *hour*, const int *minute*, const int *second*)

Checks if a UTC timestamp is accurate. A time_t timestamp is computed from the supplied datetime elements, and compared to the supplied time_t timestamp. The difference between the two, in seconds, is stored in the supplied secs_diff argument. This function is needed because the methodology used to calculate a timestamp by this library can sometimes be inaccurate when leap seconds or other unpredictable calendar changes occur. We therefore need a method to check if the returned timestamp is accurate. Other functions provided in this library call this function, so the user should not normally need to call it.

Parameters

<i>check_time</i>	The time_t timestamp to check
<i>secs_diff</i>	Modified to contain the difference, in seconds
<i>year</i>	The year
<i>month</i>	The month, 1 to 12
<i>day</i>	The day, 1 to 31, depending on the month
<i>hour</i>	The hour, 0 to 23
<i>minute</i>	The minute, 0 to 59
<i>second</i>	The second, 0 to 59

Returns

true if the supplied timestamp is accurate, false otherwise

Exceptions

bad_time	if the current time cannot be obtained.
--------------------------	---

6.1.1.2 `time_t utctime::get_day_diff ()`

Returns a `time_t` interval representing one day. The C standard does not define the units in which a `time_t` value is measured. On POSIX-compliant systems it is measured in seconds, but we cannot assume this for full portability.

Returns

A `time_t` interval representing one day.

Exceptions

<code>bad_time</code>	if the current time cannot be obtained.
---------------------------------------	---

6.1.1.3 `time_t utctime::get_hour_diff ()`

Returns a `time_t` interval representing one hour. The C standard does not define the units in which a `time_t` value is measured. On POSIX-compliant systems it is measured in seconds, but we cannot assume this for full portability.

Returns

A `time_t` interval representing one hour.

Exceptions

<code>bad_time</code>	if the current time cannot be obtained.
---------------------------------------	---

6.1.1.4 `time_t utctime::get_sec_diff ()`

Returns a `time_t` interval representing one second. The C standard does not define the units in which a `time_t` value is measured. On POSIX-compliant systems it is measured in seconds, but we cannot assume this for full portability.

Returns

A `time_t` interval representing one second.

Exceptions

<code>bad_time</code>	if the current time cannot be obtained.
---------------------------------------	---

6.1.1.5 `time_t utctime::get_utc_timestamp (const int year, const int month, const int day, const int hour, const int minute, const int second)`

Gets a `time_t` timestamp for a requested UTC time.

Parameters

<i>year</i>	The year
<i>month</i>	The month, 1 to 12
<i>day</i>	The day, 1 to 31, depending on the month
<i>hour</i>	The hour, 0 to 23
<i>minute</i>	The minute, 0 to 59
<i>second</i>	The second, 0 to 59. Leap seconds are not supported.

Returns

A `time_t` timestamp for the requested UTC time.

Exceptions

<code>bad_time</code>	if the current time cannot be obtained.
---------------------------------------	---

6.1.1.6 `int utctime::get_utc_timestamp_sec_diff (const time_t check_time, const int year, const int month, const int day, const int hour, const int minute, const int second)`

Checks a `time_t` timestamp against a UTC time, and returns the difference in seconds. This function only returns a good value if the timestamp is less than 24 hours away from the desired time, so the caller is responsible for making sure that it is. This function may also return a bad value if a leap second or other unpredictable calendar change falls between the desired UTC time and the provided time stamp. The result should therefore always be checked with `utctime::check_utc_timestamp()`, or by calling this function again.

Parameters

<i>check_time</i>	The <code>time_t</code> timestamp to check
<i>year</i>	The year
<i>month</i>	The month, 1 to 12
<i>day</i>	The day, 1 to 31, depending on the month
<i>hour</i>	The hour, 0 to 23
<i>minute</i>	The minute, 0 to 59
<i>second</i>	The second, 0 to 59. Leap seconds are not supported.

Returns

The difference, if any, represented in seconds.

Exceptions

<code>bad_time</code>	if the current time cannot be obtained.
---------------------------------------	---

6.1.1.7 `bool utctime::is_leap_year (const int year)`

Checks if the supplied year is a leap year.

Parameters

<i>year</i>	A year
-------------	--------

Returns

`true` is `year` is a leap year, `false` otherwise.

6.1.1.8 `int utctime::tm.compare (const std::tm & first, const std::tm & second)`

Compares two `std::tm` structs. Only the year, month, day, hour, minute and second are compared. Any timezone or DST information is ignored.

Parameters

<i>first</i>	The first std::tm struct.
<i>second</i>	The second std::tm struct.

Returns

-1 if *first* is earlier than *second*, 1 if *first* is later than *second*, and 0 if *first* is equal to *second*.

6.1.1.9 std::tm * utctime::tm_decrement_day (std::tm * *changing_tm*, const int *quantity* = 1)

Deducts one or more days from a std::tm struct, decrementing the month and/or the year as necessary.

Parameters

<i>changing_tm</i>	A pointer to the std::tm struct to increment. The struct referred to by the pointer is modified by the function.
<i>quantity</i>	The number of days to deduct

Returns

A pointer to the same std::tm struct.

6.1.1.10 std::tm * utctime::tm_decrement_hour (std::tm * *changing_tm*, const int *quantity* = 1)

Deducts one or more hours from a std::tm struct, decrementing the day, month and/or the year as necessary.

Parameters

<i>changing_tm</i>	A pointer to the std::tm struct to increment. The struct referred to by the pointer is modified by the function.
<i>quantity</i>	The number of hours to deduct

Returns

A pointer to the same std::tm struct.

6.1.1.11 std::tm * utctime::tm_decrement_minute (std::tm * *changing_tm*, const int *quantity* = 1)

Deducts one or more minutes from a std::tm struct, decrementing the hour, day, month and/or the year as necessary.

Parameters

<i>changing_tm</i>	A pointer to the std::tm struct to increment. The struct referred to by the pointer is modified by the function.
<i>quantity</i>	The number of minutes to deduct

Returns

A pointer to the same `std::tm` struct.

6.1.1.12 `std::tm * utctime::tm_decrement_second (std::tm * changing_tm, const int quantity = 1)`

Deducts one or more seconds from a `std::tm` struct, decrementing the minute, hour, day, month and/or the year as necessary.

Parameters

<i>changing_tm</i>	A pointer to the <code>std::tm</code> struct to increment. The struct referred to by the pointer is modified by the function.
<i>quantity</i>	The number of seconds to deduct

Returns

A pointer to the same `std::tm` struct.

6.1.1.13 `std::tm * utctime::tm_increment_day (std::tm * changing_tm, const int quantity = 1)`

Adds one or more days to a `std::tm` struct, incrementing the month and/or the year as necessary.

Parameters

<i>changing_tm</i>	A pointer to the <code>std::tm</code> struct to increment. The struct referred to by the pointer is modified by the function.
<i>quantity</i>	The number of days to add

Returns

A pointer to the same `std::tm` struct.

6.1.1.14 `std::tm * utctime::tm_increment_hour (std::tm * changing_tm, const int quantity = 1)`

Adds one or more hours to a `std::tm` struct, incrementing the day, month and/or the year as necessary.

Parameters

<i>changing_tm</i>	A pointer to the <code>std::tm</code> struct to increment. The struct referred to by the pointer is modified by the function.
<i>quantity</i>	The number of hours to add

Returns

A pointer to the same `std::tm` struct.

6.1.1.15 `std::tm * utctime::tm_increment_minute (std::tm * changing_tm, const int quantity = 1)`

Adds one or more minutes to a `std::tm` struct, incrementing the hour, day, month and/or the year as necessary.

Parameters

<i>changing_tm</i>	A pointer to the <code>std::tm</code> struct to increment. The struct referred to by the pointer is modified by the function.
<i>quantity</i>	The number of minutes to add

Returns

A pointer to the same `std::tm` struct.

6.1.1.16 `std::tm * utctime::tm_increment_second (std::tm * changing_tm, const int quantity = 1)`

Adds one or mor seconds to a `std::tm` struct, incrementing the minute, hour, day, month and/or the year as necessary.

Parameters

<i>changing_tm</i>	A pointer to the <code>std::tm</code> struct to increment. The struct referred to by the pointer is modified by the function.
<i>quantity</i>	The number of seconds to add

Returns

A pointer to the same `std::tm` struct.

6.1.1.17 `int utctime::tm_intraday_secs_diff (const std::tm & first, const std::tm & second)`

Returns the difference between two `std::tm` structs. The structs are assumed to be within 24 hours of each other, and if they are not, the returned result is computed as if they were. For instance, comparing 10:00 on one day to 14:00 on the next day will yield a difference equivalent to 4 hours, not to 28 hours.

Parameters

<i>first</i>	The first <code>std::tm</code> struct
<i>second</i>	The second <code>std::tm</code> struct

Returns

The difference, in seconds, between the two `std::tm` structs. The difference is positive if *first* is earlier than *second*, and negative if *second* is earlier than *first*.

6.1.1.18 `bool utctime::validate_date (const int year, const int month, const int day, const int hour, const int minute, const int second)`

Checks whether a supplied date is valid.

Parameters

<i>year</i>	The year
<i>month</i>	The month, 1 to 12
<i>day</i>	The day, 1 to 31, depending on the month
<i>hour</i>	The hour, 0 to 23
<i>minute</i>	The minute, 0 to 59
<i>second</i>	The second, 0 to 59. Leap seconds are not supported.

Returns

true if the date if valid.

Exceptions

<i>invalid_date</i>	if the date is not valid.
-------------------------------------	---------------------------

Todo Why does this throw an exception? Why not return false? In order so that the error message can show why it's invalid?

Chapter 7

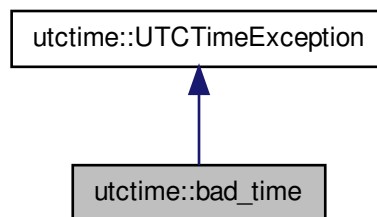
Class Documentation

7.1 utctime::bad_time Class Reference

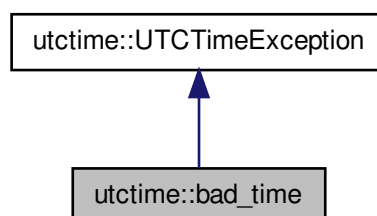
Thrown when the time cannot be obtained.

```
#include <utctime.h>
```

Inheritance diagram for utctime::bad_time:



Collaboration diagram for utctime::bad_time:



Public Member Functions

- [bad_time \(\)](#)
Constructor.

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

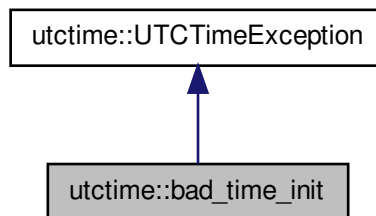
- [utctime.h](#)

7.2 utctime::bad_time_init Class Reference

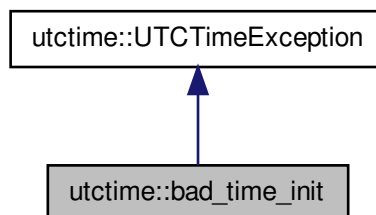
Thrown when the time cannot be initialized.

```
#include <utctime.h>
```

Inheritance diagram for utctime::bad_time_init:



Collaboration diagram for utctime::bad_time_init:



Public Member Functions

- [bad_time_init \(\)](#)
Constructor.

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

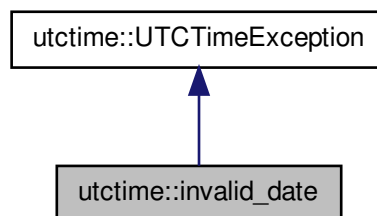
- [utctime.h](#)

7.3 utctime::invalid_date Class Reference

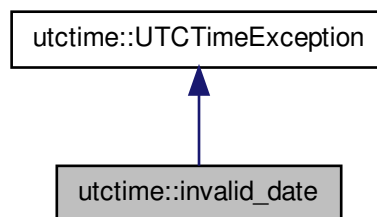
Thrown when an invalid date is provided to the constructor.

```
#include <utctime.h>
```

Inheritance diagram for utctime::invalid_date:



Collaboration diagram for utctime::invalid_date:



Public Member Functions

- [invalid_date](#) (const std::string &msg)

Constructor.

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

- [utctime.h](#)

7.4 utctime::UTCTime Class Reference

```
#include <utctime.h>
```

Public Member Functions

- [UTCTime](#) ()
Default constructor.
- [UTCTime](#) (const std::tm &utc_tm)
Constructor taking a std::tm struct.
- [UTCTime](#) (const int year, const int month, const int day, const int hour, const int minute, const int second)
Constructor taking individual date values.
- std::tm [get_tm](#) () const
Returns a std::tm struct containing the UTC datetime.
- std::string [time_string](#) () const
Returns a std::string containing the UTC datetime.
- std::string [time_string_inet](#) () const
Returns a std::string containing the UTC datetime in RFC3339 format.
- time_t [timestamp](#) () const
Returns a time_t timestamp for the UTC datetime.
- bool [operator<](#) (const [UTCTime](#) &rhs) const
Overloaded less than operator.
- bool [operator>=](#) (const [UTCTime](#) &rhs) const
Overloaded greater than or equal to operator.
- bool [operator>](#) (const [UTCTime](#) &rhs) const
Overloaded greater than operator.
- bool [operator<=](#) (const [UTCTime](#) &rhs) const
Overloaded less than or equal to operator.
- bool [operator==](#) (const [UTCTime](#) &rhs) const
Overloaded equality operator.
- bool [operator!=](#) (const [UTCTime](#) &rhs) const
Overloaded inequality operator.
- double [operator-](#) (const [UTCTime](#) &rhs) const
Overloaded subtraction operator.

7.4.1 Detailed Description

A class for holding a UTC time.

7.4.2 Constructor & Destructor Documentation

7.4.2.1 [UTCTime::UTCTime](#) () [explicit]

Default constructor, which initializes to the current time.

Exceptions

bad_time_init	if unable to get the current time.
-------------------------------	------------------------------------

7.4.2.2 [UTCTime::UTCTime](#) (const std::tm & utc_tm) [explicit]

Constructor taking a std::tm struct.

Parameters

<i>utc_tm</i>	A std::tm struct containing the desired initialization time.
---------------	--

Exceptions

<i>bad_time_init</i>	if unable to get the current time.
<i>invalid_date</i>	if the supplied date is bad.

7.4.2.3 `UTCTime::UTCTime (const int year, const int month, const int day, const int hour, const int minute, const int second) [explicit]`

Constructor taking individual date values.

Parameters

<i>year</i>	The year
<i>month</i>	The month, 1 to 12
<i>day</i>	The day, 1 to 31, depending on month
<i>hour</i>	The hour, 0 to 23
<i>minute</i>	The minute, 0 to 59
<i>second</i>	The second, 0 to 59. Leap seconds are not supported.

Exceptions

<i>bad_time_init</i>	if unable to get the current time.
<i>invalid_date</i>	if the supplied date is bad.

7.4.3 Member Function Documentation

7.4.3.1 `std::tm UTCTime::get_tm () const`

Returns a std::tm struct containing the UTC datetime.

Returns

A std::tm struct containing the UTC datetime.

7.4.3.2 `bool UTCTime::operator!= (const UTCTime & rhs) const`

Overloaded inequality operator

Parameters

<i>rhs</i>	The UTCTime instance to which to compare
------------	--

Returns

true if the instance is not equal to *rhs*, false otherwise.

7.4.3.3 `double UTCTime::operator- (const UTCTime & rhs) const`

Overloaded subtraction operator

Parameters

<i>rhs</i>	The UTCTime instance to subtract from the instance
------------	--

Returns

The difference, in seconds, between the two instances.

7.4.3.4 `bool UTCTime::operator< (const UTCTime & rhs) const`

Overloaded less than operator

Parameters

<i>rhs</i>	The UTCTime instance to which to compare
------------	--

Returns

true if the instance is less than `rhs`, false otherwise.

7.4.3.5 `bool UTCTime::operator<= (const UTCTime & rhs) const`

Overloaded less than or equal to operator

Parameters

<i>rhs</i>	The UTCTime instance to which to compare
------------	--

Returns

true if the instance is less than or equal to `rhs`, false otherwise.

7.4.3.6 `bool UTCTime::operator== (const UTCTime & rhs) const`

Overloaded equality operator

Parameters

<i>rhs</i>	The UTCTime instance to which to compare
------------	--

Returns

true if the instance is equal to `rhs`, false otherwise.

7.4.3.7 `bool UTCTime::operator> (const UTCTime & rhs) const`

Overloaded greater than operator

Parameters

<i>rhs</i>	The UTCTime instance to which to compare
------------	--

Returns

true if the instance is greater than `rhs`, false otherwise.

7.4.3.8 `bool UTCTime::operator>= (const UTCTime & rhs) const`

Overloaded greater than or equal to operator

Parameters

<code>rhs</code>	The UTCTime instance to which to compare
------------------	--

Returns

true if the instance is greater than or equal to `rhs`, false otherwise.

7.4.3.9 `std::string UTCTime::time_string () const`

Returns a `std::string` containing the UTC datetime.

Returns

A `std::string` containing the UTC datetime.

Exceptions

bad_time	if unable to get the time.
--------------------------	----------------------------

7.4.3.10 `std::string UTCTime::time_string_inet () const`

Returns a `std::string` containing the UTC datetime in RFC3339 format.

Returns

A `std::string` containing the UTC datetime in RFC3339 format.

Exceptions

bad_time	if unable to get the time.
--------------------------	----------------------------

7.4.3.11 `time_t UTCTime::timestamp () const`

Returns a `time_t` timestamp for the UTC datetime.

Returns

A `time_t` timestamp for the UTC datetime.

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

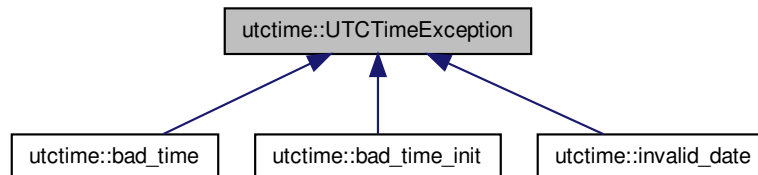
- [utctime.h](#)
- [utctime.cpp](#)

7.5 utctime::UTCTimeException Class Reference

Base exception class.

```
#include <utctime.h>
```

Inheritance diagram for utctime::UTCTimeException:



Public Member Functions

- [UTCTimeException](#) (const std::string msg="No error message")
Constructor.
- virtual [~UTCTimeException](#) ()
Destructor.
- const std::string & [what](#) () const
Returns an error message.

7.5.1 Detailed Description

All custom exceptions thrown by the `UTCTime()` class and associated functions are derived from [UTCTimeException](#).

7.5.2 Constructor & Destructor Documentation

7.5.2.1 `utctime::UTCTimeException::UTCTimeException (const std::string msg = "No error message")`
[inline], [explicit]

Parameters

<i>msg</i>	An error message to print when what () is called.
------------	---

7.5.3 Member Function Documentation

7.5.3.1 `const std::string& utctime::UTCTimeException::what () const` [inline]

Returns

A std::string containing the error message.

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

- [utctime.h](#)

Chapter 8

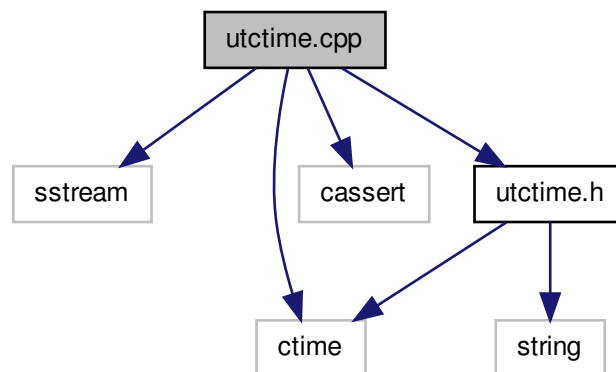
File Documentation

8.1 utctime.cpp File Reference

Implementation of UTCTime() class and associated functions.

```
#include <sstream>
#include <ctime>
#include <cassert>
#include "utctime.h"
```

Include dependency graph for utctime.cpp:



8.1.1 Detailed Description

Implementation of UTCTime() class and associated functions.

Author

Paul Griffiths

Copyright

Copyright 2013 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

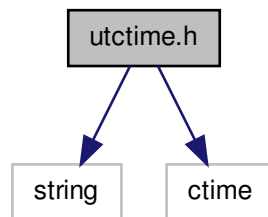
8.2 utctime.h File Reference

Interface to UTCTime() class and associated functions.

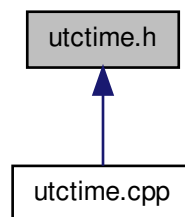
```
#include <string>
```

```
#include <ctime>
```

Include dependency graph for utctime.h:



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



Classes

- class [utctime::UTCTimeException](#)
Base exception class.
- class [utctime::bad_time_init](#)
Thrown when the time cannot be initialized.
- class [utctime::bad_time](#)
Thrown when the time cannot be obtained.
- class [utctime::invalid_date](#)
Thrown when an invalid date is provided to the constructor.
- class [utctime::UTCTime](#)

Namespaces

- namespace [utctime](#)

Namespace for UTCTime() class and associated functions.

Functions

- bool `utctime::validate_date` (const int year, const int month, const int day, const int hour, const int minute, const int second)
Checks whether a supplied date is valid.
- time_t `utctime::get_day_diff` ()
Returns a time_t interval representing one day.
- time_t `utctime::get_hour_diff` ()
Returns a time_t interval representing one hour.
- time_t `utctime::get_sec_diff` ()
Returns a time_t interval representing one second.
- int `utctime::tm_compare` (const std::tm &first, const std::tm &second)
Compares two std::tm structs.
- int `utctime::tm_intraday_secs_diff` (const std::tm &first, const std::tm &second)
Returns the difference between two std::tm structs.
- bool `utctime::is_leap_year` (const int year)
Checks if the supplied year is a leap year.
- std::tm * `utctime::tm_increment_day` (std::tm *changing_tm, const int quantity=1)
Adds one or more days to a std::tm struct.
- std::tm * `utctime::tm_increment_hour` (std::tm *changing_tm, const int quantity=1)
Adds one or more hours to a std::tm struct.
- std::tm * `utctime::tm_increment_minute` (std::tm *changing_tm, const int quantity=1)
Adds one or more minutes to a std::tm struct.
- std::tm * `utctime::tm_increment_second` (std::tm *changing_tm, const int quantity=1)
Adds one or more seconds to a std::tm struct.
- std::tm * `utctime::tm_decrement_day` (std::tm *changing_tm, const int quantity=1)
Deducts one or more days from a std::tm struct.
- std::tm * `utctime::tm_decrement_hour` (std::tm *changing_tm, const int quantity=1)
Deducts one or more hours from a std::tm struct.
- std::tm * `utctime::tm_decrement_minute` (std::tm *changing_tm, const int quantity=1)
Deducts one or more minutes from a std::tm struct.
- std::tm * `utctime::tm_decrement_second` (std::tm *changing_tm, const int quantity=1)
Deducts one or more seconds from a std::tm struct.
- bool `utctime::check_utc_timestamp` (const time_t check_time, int &secs_diff, const int year, const int month, const int day, const int hour, const int minute, const int second)
Checks if a UTC timestamp is accurate.
- time_t `utctime::get_utc_timestamp` (const int year, const int month, const int day, const int hour, const int minute, const int second)
Gets a time_t timestamp for a requested UTC time.
- int `utctime::get_utc_timestamp_sec_diff` (const time_t check_time, const int year, const int month, const int day, const int hour, const int minute, const int second)
Checks a time_t timestamp against a UTC time.

8.2.1 Detailed Description

Interface to UTCTime() class and associated functions.

Author

Paul Griffiths

Copyright

Copyright 2013 Paul Griffiths. Distributed under the terms of the GNU General Public License. <http://www.gnu.org/licenses/>

Index

- check_utc_timestamp
 - utctime, [12](#)
- get_day_diff
 - utctime, [12](#)
- get_hour_diff
 - utctime, [13](#)
- get_sec_diff
 - utctime, [13](#)
- get_tm
 - utctime::UTCTime, [23](#)
- get_utc_timestamp
 - utctime, [13](#)
- get_utc_timestamp_sec_diff
 - utctime, [14](#)
- is_leap_year
 - utctime, [14](#)
- operator<
 - utctime::UTCTime, [24](#)
- operator<=
 - utctime::UTCTime, [24](#)
- operator>
 - utctime::UTCTime, [24](#)
- operator>=
 - utctime::UTCTime, [25](#)
- operator-
 - utctime::UTCTime, [23](#)
- operator==
 - utctime::UTCTime, [24](#)
- time_string
 - utctime::UTCTime, [25](#)
- time_string_inet
 - utctime::UTCTime, [25](#)
- timestamp
 - utctime::UTCTime, [25](#)
- tm_compare
 - utctime, [14](#)
- tm_decrement_day
 - utctime, [15](#)
- tm_decrement_hour
 - utctime, [15](#)
- tm_decrement_minute
 - utctime, [15](#)
- tm_decrement_second
 - utctime, [16](#)
- tm_increment_day
 - utctime, [16](#)
- tm_increment_hour
 - utctime, [16](#)
- tm_increment_minute
 - utctime, [16](#)
- tm_increment_second
 - utctime, [17](#)
- tm_intraday_secs_diff
 - utctime, [17](#)
- UTCTime
 - utctime::UTCTime, [22, 23](#)
- UTCTimeException
 - utctime::UTCTimeException, [26](#)
- utctime, [11](#)
 - check_utc_timestamp, [12](#)
 - get_day_diff, [12](#)
 - get_hour_diff, [13](#)
 - get_sec_diff, [13](#)
 - get_utc_timestamp, [13](#)
 - get_utc_timestamp_sec_diff, [14](#)
 - is_leap_year, [14](#)
 - tm_compare, [14](#)
 - tm_decrement_day, [15](#)
 - tm_decrement_hour, [15](#)
 - tm_decrement_minute, [15](#)
 - tm_decrement_second, [16](#)
 - tm_increment_day, [16](#)
 - tm_increment_hour, [16](#)
 - tm_increment_minute, [16](#)
 - tm_increment_second, [17](#)
 - tm_intraday_secs_diff, [17](#)
 - validate_date, [17](#)
- utctime.cpp, [27](#)
- utctime.h, [28](#)
- utctime::UTCTime, [21](#)
 - get_tm, [23](#)
 - operator<, [24](#)
 - operator<=, [24](#)
 - operator>, [24](#)
 - operator>=, [25](#)
 - operator-, [23](#)
 - operator==, [24](#)
 - time_string, [25](#)
 - time_string_inet, [25](#)
 - timestamp, [25](#)
 - UTCTime, [22, 23](#)
- utctime::UTCTimeException, [26](#)
 - UTCTimeException, [26](#)
 - what, [26](#)
- utctime::bad_time, [19](#)

utctime::bad_time_init, [20](#)

utctime::invalid_date, [21](#)

validate_date

utctime, [17](#)

what

utctime::UTCTimeException, [26](#)