cpptoolkit

Generated by Doxygen 1.9.1

Chapter 1

C++ Toolkit Documentation

1.1 Description

This toolkit provides a collection of utilities and tools to enhance C++ development. It includes various modules for file handling, string manipulation, and more.

1.2 Build Process

To build the toolkit, follow these steps:

1. Ensure you have CMake installed on your system.
2. Clone the repository to your local machine.
3. Navigate to the root directory of the repository.
4. Create a build directory: `mkdir build && cd build`
5. Run CMake to configure the project: `cmake ..`
6. Build the project using Make: `make`
7. The compiled binaries will be located in the `build` directory.

Chapter 2

Hierarchical 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 documented files with brief descriptions:

Chapter 5

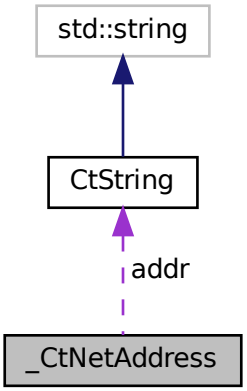
Class Documentation

5.1 _CtNetAddress Struct Reference

Struct describing a network address.

`#include <CtTypes.hpp>`

Collaboration diagram for _CtNetAddress:



Public Attributes

- CtString **addr**
- CtUInt16 **port**

5.1.1 Detailed Description

Struct describing a network address.

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

- include/CtTypes.hpp

5.2 _CtNetMessage Struct Reference

Struct describing data sent over network.

`#include <CtTypes.hpp>`

Public Attributes

- CtUInt8 **data** [CTNET_BUFFER_SIZE]
- CtUInt32 **size** = CTNET_BUFFER_SIZE

5.2.1 Detailed Description

Struct describing data sent over network.

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

- include/CtTypes.hpp

5.3 CtConfig Class Reference

A configuration file parser class for extracting various data types from configuration values.

`#include <CtConfig.hpp>`

Public Member Functions

- EXPORTED_API CtConfig (const std::string &p_configFile)

  Constructor for CtConfig.
- EXPORTED_API ~CtConfig ()

  Destructor for cleaning up resources.
- EXPORTED_API void read ()

  Read data from config file. This method can throw CtFileParseError if file cannot be parsed. This method can throw CtFileError if there is a problem with the file.
- EXPORTED_API void write ()

  Write data to config file.
- EXPORTED_API int32_t parseAsInt (const std::string &p_key)

  Parse a value as a 32-bit signed integer or throw CtKeyNotFoundError if key is not found in the map or throw CtParseError if key value cannot be parsed as int.
- EXPORTED_API uint32_t parseAsUInt (const std::string &p_key)

  Parse a value as a 32-bit unsigned integer or throw CtKeyNotFoundError if key is not found in the map or throw CtParseError if key value cannot be parsed as uint.
- EXPORTED_API float parseAsFloat (const std::string &p_key)

  Parse a value as a float or throw CtKeyNotFoundError if key is not found in the map or throw CtParseError if key value cannot be parsed as float.
- EXPORTED_API double parseAsDouble (const std::string &p_key)

  Parse a value as a double-precision floating-point number or throw CtKeyNotFoundError if key is not found in the map or throw CtParseError if key value cannot be parsed as double.

- EXPORTED_API std::string parseAsString (const std::string &p_key)

  Parse a value as a standard C++ string or throw CtKeyNotFoundError if key is not found in the map.
- EXPORTED_API void writeInt (const std::string &p_key, const int32_t &p_value)

  Write value to key as int.
- EXPORTED_API void writeUInt (const std::string &p_key, const uint32_t &p_value)

  Write value to key as uint.
- EXPORTED_API void writeFloat (const std::string &p_key, const float &p_value)

  Write value to key as float.
- EXPORTED_API void writeDouble (const std::string &p_key, const double &p_value)

  Write value to key as double.
- EXPORTED_API void writeString (const std::string &p_key, const std::string &p_value)

  Write value to key as string.

5.3.1 Detailed Description

A configuration file parser class for extracting various data types from configuration values.

5.3.2 Constructor & Destructor Documentation

5.3.2.1 CtConfig()

```
CtConfig::CtConfig (
            const std::string & p_configFile )
```

Constructor for CtConfig.

Parameters

configFile	The path to the configuration file to be parsed.

5.3.3 Member Function Documentation

5.3.3.1 parseAsDouble()

```
double CtConfig::parseAsDouble (
            const std::string & p_key )
```

Parse a value as a double-precision floating-point number or throw CtKeyNotFoundError if key is not found in the map or throw CtParseError if key value cannot be parsed as double.

Parameters

key	The key value to be parsed.

Returns

The parsed double value.

5.3.3.2 parseAsFloat()

```
float CtConfig::parseAsFloat (
            const std::string & p_key )
```

Parse a value as a float or throw CtKeyNotFoundError if key is not found in the map or throw CtParseError if key value cannot be parsed as float.

Parameters

key	The key value to be parsed.

Returns

The parsed floating-point value.

5.3.3.3 parseAsInt()

```
int32_t CtConfig::parseAsInt (
            const std::string & p_key )
```

Parse a value as a 32-bit signed integer or throw CtKeyNotFoundError if key is not found in the map or throw CtParseError if key value cannot be parsed as int.

Parameters

key	The key value to be parsed.

Returns

The parsed integer value.

5.3.3.4 parseAsString()

```
std::string CtConfig::parseAsString (
            const std::string & p_key )
```

Parse a value as a standard C++ string or throw CtKeyNotFoundError if key is not found in the map.

Parameters

key	The key value to be parsed.

Returns

The parsed string.

5.3.3.5 parseAsUInt()

```
uint32_t CtConfig::parseAsUInt (
            const std::string & p_key )
```

Parse a value as a 32-bit unsigned integer or throw CtKeyNotFoundError if key is not found in the map or throw CtParseError if key value cannot be parsed as uint.

Parameters

key	The key value to be parsed.

Returns

The parsed unsigned integer value.

5.3.3.6 writeDouble()

```
void CtConfig::writeDouble (
            const std::string & p_key,
            const double & p_value )
```

Write value to key as double.

Parameters

p_key	The key value.
p_value	The value to be written for this key.

5.3.3.7 writeFloat()

```
void CtConfig::writeFloat (
            const std::string & p_key,
            const float & p_value )
```

Write value to key as float.

Parameters

p_key	The key value.
p_value	The value to be written for this key.

5.3.3.8 writeInt()

```
void CtConfig::writeInt (
            const std::string & p_key,
            const int32_t & p_value )
```

Write value to key as int.

Parameters

p_key	The key value.
p_value	The value to be written for this key.

5.3.3.9 writeString()

```
void CtConfig::writeString (
            const std::string & p_key,
            const std::string & p_value )
```

Write value to key as string.

Parameters

p_key	The key value.
p_value	The value to be written for this key.

5.3.3.10 writeUInt()

```
void CtConfig::writeUInt (
            const std::string & p_key,
            const uint32_t & p_value )
```

Write value to key as uint.

Parameters

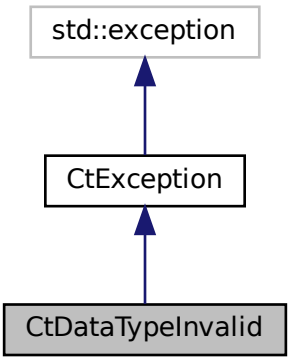
p_key	The key value.
p_value	The value to be written for this key.

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

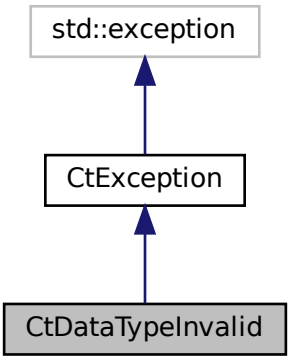
- include/utils/CtConfig.hpp
- src/utils/CtConfig.cpp

5.4 CtDataTypeInvalid Class Reference

Inheritance diagram for CtDataTypeInvalid:



Collaboration diagram for CtDataTypeInvalid:



Public Member Functions

- **CtDataTypeInvalid** (const std::string &msg)

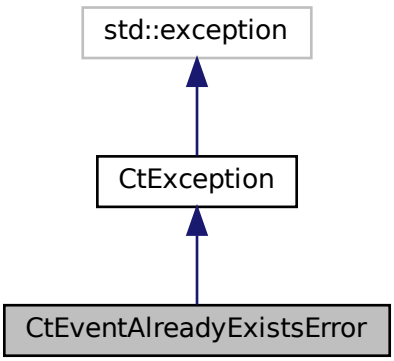
Additional Inherited Members

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

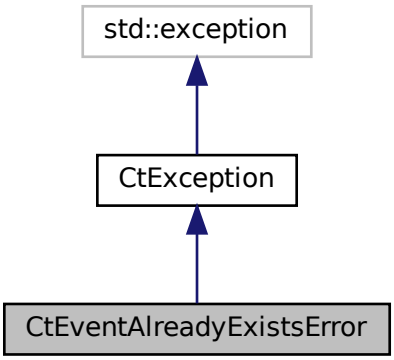
- include/exceptions/CtTypeExceptions.hpp

5.5 CtEventAlreadyExistsError Class Reference

Inheritance diagram for CtEventAlreadyExistsError:



Collaboration diagram for CtEventAlreadyExistsError:



Public Member Functions

- **CtEventAlreadyExistsError** (const std::string &msg)

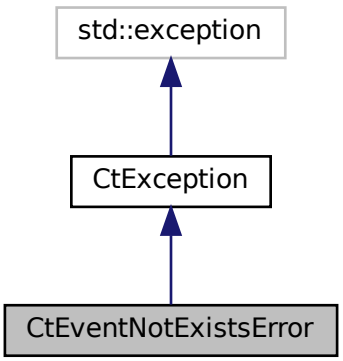
Additional Inherited Members

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

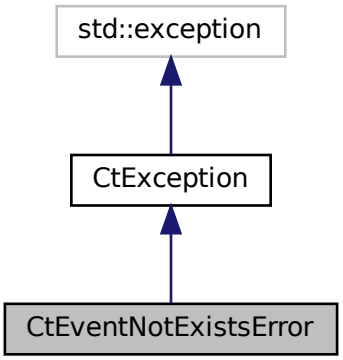
- include/exceptions/CtEventExceptions.hpp

5.6 CtEventNotExistsError Class Reference

Inheritance diagram for CtEventNotExistsError:



Collaboration diagram for CtEventNotExistsError:



Public Member Functions

- **CtEventNotExistsError** (const std::string &msg)

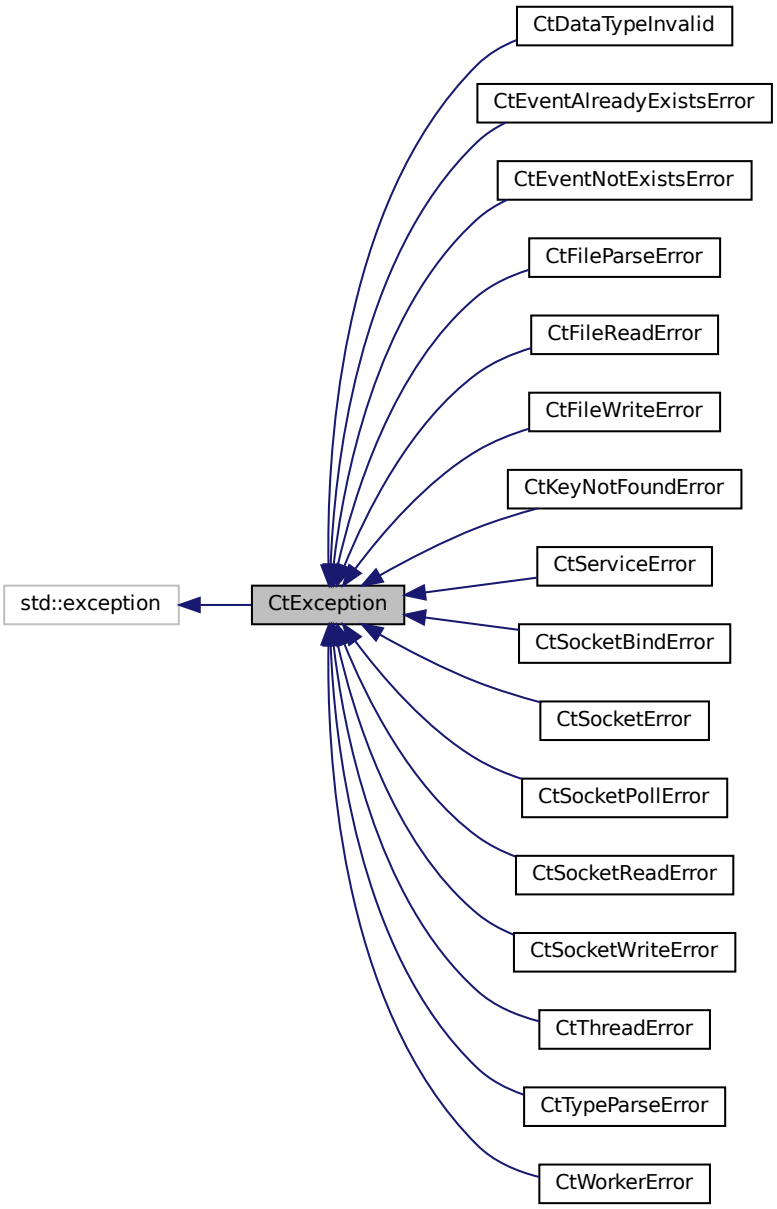
Additional Inherited Members

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

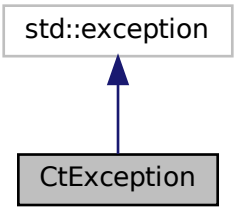
- include/exceptions/CtEventExceptions.hpp

5.7 CtException Class Reference

Inheritance diagram for CtException:



Collaboration diagram for CtException:



Public Member Functions

- const char * **what** () const noexcept override

Protected Member Functions

- **CtException** (const std::string &msg)

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

- include/exceptions/CtException.hpp

5.8 CtFileInput Class Reference

Public Member Functions

- EXPORTED_API CtFileInput (const std::string &p_fileName)
  Constructs the CtFileInput object.
- EXPORTED_API ~CtFileInput ()
  Destructor for CtFileInput.
- EXPORTED_API void setDelimiter (const char *p_delim, CtUInt8 p_delim_size)
  Set the the delimiter of read() method.
- EXPORTED_API bool read (CtRawData *p_data)
  This method read data from the file.

5.8.1 Constructor & Destructor Documentation

5.8.1.1 CtFileInput()

```
CtFileInput::CtFileInput (
            const std::string & p_fileName )
```

Constructs the CtFileInput object.

Parameters

p_fileName	Filename.

5.8.1.2 ~CtFileInput()

```
CtFileInput::~CtFileInput ( )
```

Destructor for CtFileInput.

Performs any necessary cleanup.

5.8.2 Member Function Documentation

5.8.2.1 read()

```
bool CtFileInput::read (
            CtRawData * p_data )
```

This method read data from the file.

Parameters

p_data	Where to store the data read

Returns

bool Returns True on success or False on EOF.

5.8.2.2 setDelimiter()

```
void CtFileInput::setDelimiter (
            const char * p_delim,
            CtUInt8 p_delim_size )
```

Set the the delimiter of read() method.

Parameters

p_delim	The delimiter.
p_delim_size	The delimiter size.

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

- include/io/CtFileInput.hpp
- src/io/CtFileInput.cpp

5.9 CtFileOutput Class Reference

Public Types

- enum class WriteMode { **Append** , **Truncate** }

  Enum representing write mode.

Public Member Functions

- EXPORTED_API CtFileOutput (const std::string &p_fileName, WriteMode p_mode=WriteMode::Append)

  Constructs the CtFileOutput object.
- EXPORTED_API ~CtFileOutput ()

  Destructor for CtFileOutput.
- EXPORTED_API void setDelimiter (const char *p_delim, CtUInt8 p_delim_size)

  Set the the delimiter of write() method.
- EXPORTED_API void write (CtRawData *p_data)

  This method writes to file.

5.9.1 Constructor & Destructor Documentation

5.9.1.1 CtFileOutput()

```
CtFileOutput::CtFileOutput (
            const std::string & p_fileName,
            WriteMode p_mode = WriteMode::Append )
```

Constructs the CtFileOutput object.

Parameters

p_fileName	Filename.

5.9.1.2 ~CtFileOutput()

```
CtFileOutput::~CtFileOutput ( )
```

Destructor for CtFileOutput.

Performs any necessary cleanup.

5.9.2 Member Function Documentation

5.9.2.1 setDelimiter()

```
void CtFileOutput::setDelimiter (
            const char * p_delim,
            CtUInt8 p_delim_size )
```

Set the the delimiter of write() method.

Parameters

p_delim	The delimiter.
p_delim_size	The delimiter size.

5.9.2.2 write()

```
void CtFileOutput::write (
            CtRawData * p_data )
```

This method writes to file.

Parameters

p_data	The data to be written.

Returns

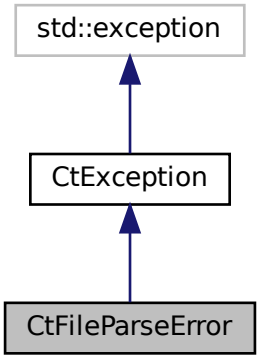
void

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

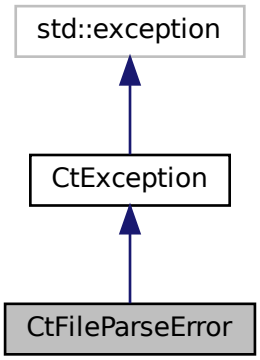
- include/io/CtFileOutput.hpp
- src/io/CtFileOutput.cpp

5.10 CtFileParseError Class Reference

Inheritance diagram for CtFileParseError:



Collaboration diagram for CtFileParseError:



Public Member Functions

- **CtFileParseError** (const std::string &msg)

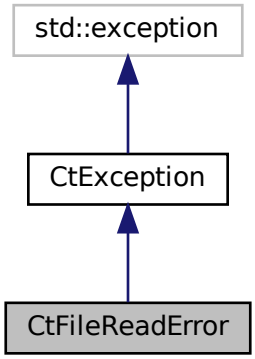
Additional Inherited Members

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

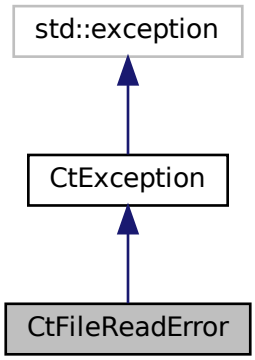
- include/exceptions/CtFileExceptions.hpp

5.11 CtFileReadError Class Reference

Inheritance diagram for CtFileReadError:



Collaboration diagram for CtFileReadError:



Public Member Functions

- **CtFileReadError** (const std::string &msg)

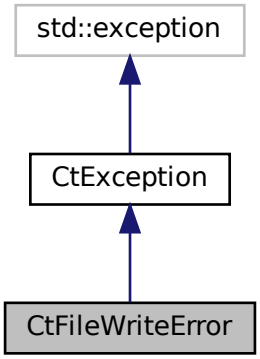
Additional Inherited Members

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

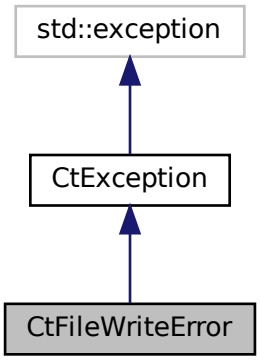
- include/exceptions/CtFileExceptions.hpp

5.12 CtFileWriteError Class Reference

Inheritance diagram for CtFileWriteError:



Collaboration diagram for CtFileWriteError:



Public Member Functions

- **CtFileWriteError** (const std::string &msg)

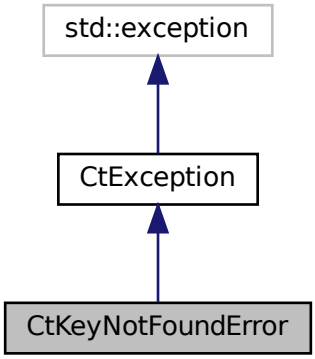
Additional Inherited Members

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

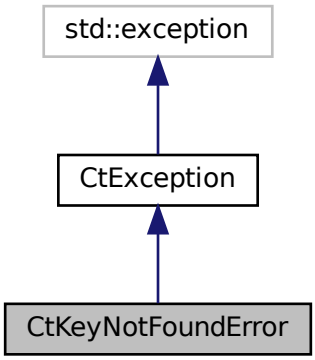
- include/exceptions/CtFileExceptions.hpp

5.13 CtKeyNotFoundError Class Reference

Inheritance diagram for CtKeyNotFoundError:



Collaboration diagram for CtKeyNotFoundError:



Public Member Functions

- **CtKeyNotFoundError** (const std::string &msg)

Additional Inherited Members

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

- include/exceptions/CtTypeExceptions.hpp

5.14 CtLogger Class Reference

A simple logger with log levels and timestamp.

`#include <CtLogger.hpp>`

Public Types

- enum class Level {
  DEBUG , **INFO** , **WARNING** , **ERROR** ,
  CRITICAL }

  Enum representing log levels.

Public Member Functions

- EXPORTED_API CtLogger (CtLogger::Level level=CtLogger::Level::DEBUG, const std::string &componentName="")

  Constructs a CtLogger with a component name.
- EXPORTED_API ~CtLogger ()

  Destructor.
- EXPORTED_API void log_debug (const std::string &message)

  Log a message with debug log level.
- EXPORTED_API void log_info (const std::string &message)

  Log a message with info log level.
- EXPORTED_API void log_warning (const std::string &message)

  Log a message with warning log level.
- EXPORTED_API void log_error (const std::string &message)

  Log a message with error log level.
- EXPORTED_API void log_critical (const std::string &message)

  Log a message with critical log level.

Static Public Member Functions

- static EXPORTED_API CtLogger::Level stringToLevel (const std::string &level_str)

  Given the logger output level in string format this method returns the enum CtLogger::Level format.

5.14.1 Detailed Description

A simple logger with log levels and timestamp.

5.14.2 Constructor & Destructor Documentation

5.14.2.1 CtLogger()

```
CtLogger::CtLogger (
            CtLogger::Level level = CtLogger::Level::DEBUG,
            const std::string & componentName = "" )  [explicit]
```

Constructs a CtLogger with a component name.

Parameters

level	The selected level given as CtLogger::Level. All messages that have level above or equal to this value will be logged.
componentName	The name of the component or module.

5.14.3 Member Function Documentation

5.14.3.1 log_critical()

```
void CtLogger::log_critical (
            const std::string & message )
```

Log a message with critical log level.

Parameters

message	The log message.

5.14.3.2 log_debug()

```
void CtLogger::log_debug (
            const std::string & message )
```

Log a message with debug log level.

Parameters

message	The log message.

5.14.3.3 log_error()

```
void CtLogger::log_error (
            const std::string & message )
```

Log a message with error log level.

Parameters

message	The log message.

5.14.3.4 log_info()

```
void CtLogger::log_info (
            const std::string & message )
```

Log a message with info log level.

Parameters

message	The log message.

5.14.3.5 log_warning()

```
void CtLogger::log_warning (
            const std::string & message )
```

Log a message with warning log level.

Parameters

message	The log message.

5.14.3.6 stringToLevel()

```
CtLogger::Level CtLogger::stringToLevel (
            const std::string & level_str )  [static]
```

Given the logger output level in string format this method returns the enum CtLogger::Level format.

Parameters

level_str	The level in string format.

Returns

CtLogger::Level The level in enum format.

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

- include/utils/CtLogger.hpp
- src/utils/CtLogger.cpp

5.15 CtObject Class Reference

This abstract class can be used as a base class for objects that can trigger events.

`#include <CtObject.hpp>`

Public Member Functions

- template<typename F , typename... FArgs>
  EXPORTED_API void connectEvent (CtUInt32 p_eventCode, F &&func, FArgs &&... fargs)

  This method connects an event code with a function that should be triggered.
- EXPORTED_API void connectEvent (CtUInt32 p_eventCode, CtTask &p_task)

  This method connects an event code with a function that should be triggered.
- EXPORTED_API void waitPendingEvents ()

  This method is equivalent to join() function of CtThread.
- template<typename F , typename... FArgs>
  void **connectEvent** (CtObject *p_obj, CtUInt32 p_eventCode, F &&func, FArgs &&... fargs)
- template<typename F , typename... FArgs>
  void **connectEvent** (CtUInt32 p_eventCode, F &&func, FArgs &&... fargs)

Static Public Member Functions

- template<typename F , typename... FArgs>
  static EXPORTED_API void connectEvent (CtObject *p_obj, CtUInt32 p_eventCode, F &&func, FArgs &&... fargs)

  This method connects an event code with a function that should be triggered.
- static EXPORTED_API void connectEvent (CtObject *p_obj, CtUInt32 p_eventCode, CtTask &p_task)

  This method connects an event code with a function that should be triggered.

Protected Member Functions

- EXPORTED_API CtObject ()

  The constructor of the CtObject class.
- EXPORTED_API ~CtObject ()

  The destructor of the CtObject class.
- EXPORTED_API void triggerEvent (CtUInt32 p_eventCode)

  This method triggers a specific event code.
- EXPORTED_API void registerEvent (CtUInt32 p_eventCode)

  This event registers a specific event code.

5.15.1 Detailed Description

This abstract class can be used as a base class for objects that can trigger events.

5.15.2 Member Function Documentation

5.15.2.1 connectEvent() [1/4]

```
void CtObject::connectEvent (
            CtObject * p_obj,
            CtUInt32 p_eventCode,
            CtTask & p_task )  [static]
```

This method connects an event code with a function that should be triggered.

Parameters

p_obj	The object that hosts the event.
p_eventCode	The event code.
p_task	The task to be executed.

5.15.2.2 connectEvent() [2/4]

```
template<typename F , typename...  FArgs>
static EXPORTED_API void CtObject::connectEvent (
            CtObject * p_obj,
            CtUInt32 p_eventCode,
            F && func,
            FArgs &&...  fargs )  [static]
```

This method connects an event code with a function that should be triggered.

Parameters

p_obj	The object that hosts the event.
p_eventCode	The event code.
func	The function to be executed.
fargs	The parameters of the function that will be executed.

5.15.2.3 connectEvent() [3/4]

```
void CtObject::connectEvent (
            CtUInt32 p_eventCode,
            CtTask & p_task )
```

This method connects an event code with a function that should be triggered.

Parameters

p_eventCode	The event code.
p_task	The task to be executed.

5.15.2.4 connectEvent() [4/4]

```
template<typename F , typename...  FArgs>
EXPORTED_API void CtObject::connectEvent (
```

```
            CtUInt32 p_eventCode,
            F && func,
            FArgs &&... fargs )
```

This method connects an event code with a function that should be triggered.

Parameters

p_eventCode	The event code.
func	The function to be executed.
fargs	The parameters of the function that will be executed.

5.15.2.5 registerEvent()

```
void CtObject::registerEvent (
            CtUInt32 p_eventCode )  [protected]
```

This event registers a specific event code.

Parameters

p_eventCode	The event code to be registered.

5.15.2.6 triggerEvent()

```
void CtObject::triggerEvent (
            CtUInt32 p_eventCode )  [protected]
```

This method triggers a specific event code.

Parameters

p_eventCode	The event code to be triggered.

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

- include/utils/CtObject.hpp
- src/utils/CtObject.cpp

5.16 CtRawData Class Reference

Struct describing raw data.

```
#include <CtTypes.hpp>
```

Public Member Functions

- EXPORTED_API **CtRawData** (CtUInt32 p_size)
- EXPORTED_API **CtRawData** (CtRawData &p_data)
- EXPORTED_API void **nextByte** (char byte)
- EXPORTED_API CtUInt8 * **getNLastBytes** (CtUInt32 p_num)
- EXPORTED_API void **removeNLastBytes** (CtUInt32 p_num)
- EXPORTED_API CtUInt32 **size** ()
- EXPORTED_API CtUInt32 **maxSize** ()
- EXPORTED_API CtUInt8 * **get** ()
- EXPORTED_API void **clone** (CtUInt8 *p_data, CtUInt32 p_size)
- EXPORTED_API void **clone** (CtRawData &p_data)
- EXPORTED_API void **reset** ()

5.16.1 Detailed Description

Struct describing raw data.

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

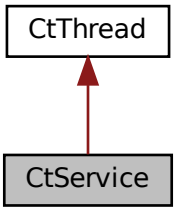
- include/CtTypes.hpp

5.17 CtService Class Reference

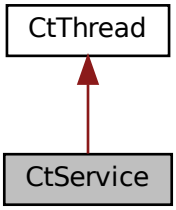
A class representing a service that runs a given task at regular intervals using a worker thread.

`#include <CtService.hpp>`

Inheritance diagram for CtService:



Collaboration diagram for CtService:



Public Member Functions

- EXPORTED_API CtService (uint64_t nslots, CtTask &task)

  Constructor for CtService.
- template<typename F , typename... FArgs>
  EXPORTED_API CtService (CtUInt32 nslots, F &&func, FArgs &&... fargs)

  Constructor for CtService.
- EXPORTED_API ~CtService ()

  Destructor for CtService.
- EXPORTED_API void runService ()

  Run the task provided by the service.
- EXPORTED_API void stopService ()

  Stop the task provided by the service.
- template<typename F , typename... FArgs>
  CtService (CtUInt32 nslots, F &&func, FArgs &&... fargs)

Static Public Attributes

- static CtUInt32 m_slot_time = 10

  The time interval for each "slot" in milliseconds.

5.17.1 Detailed Description

A class representing a service that runs a given task at regular intervals using a worker thread.

5.17.2 Constructor & Destructor Documentation

5.17.2.1 CtService() [1/2]

```
CtService::CtService (
            uint64_t nslots,
            CtTask & task )
```

Constructor for CtService.

Parameters

nslots	The time slots between task executions in milliseconds. Default is 0 (run immediately).
task	The task to be executed by the service.

5.17.2.2 CtService() [2/2]

```
template<typename F , typename...  FArgs>
EXPORTED_API CtService::CtService (
            CtUInt32 nslots,
            F && func,
            FArgs &&...  fargs )
```

Constructor for CtService.

Parameters

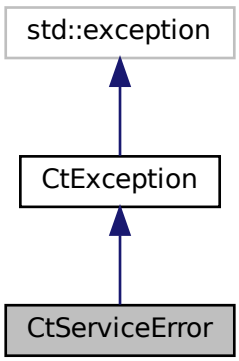
nslots	The time slots between task executions in milliseconds. Default is 0 (run immediately).
func	The task function to be executed by the service.
fargs	The task function's parameters.

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

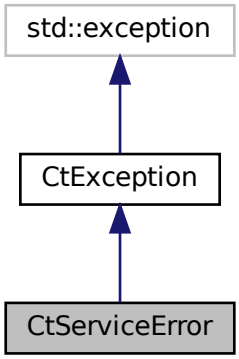
- include/threading/CtService.hpp
- src/threading/CtService.cpp

5.18 CtServiceError Class Reference

Inheritance diagram for CtServiceError:



Collaboration diagram for CtServiceError:



Public Member Functions

- **CtServiceError** (const std::string &msg)

Additional Inherited Members

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

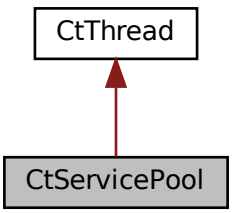
- include/exceptions/CtThreadExceptions.hpp

5.19 CtServicePool Class Reference

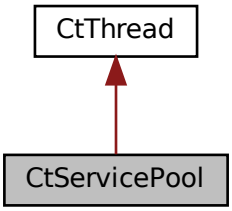
A service pool for managing and executing tasks at specified intervals using a worker pool.

`#include <CtServicePool.hpp>`

Inheritance diagram for CtServicePool:



Collaboration diagram for CtServicePool:



Public Member Functions

- EXPORTED_API CtServicePool (CtUInt32 nworkers)
  Constructor for CtServicePool.
- EXPORTED_API ~CtServicePool ()
  Destructor for CtServicePool.
- EXPORTED_API void addTask (CtUInt32 nslots, std::string id, CtTask &task)
  Add a task to the service pool with a specified interval and an optional ID.
- template<typename F , typename... FArgs>
  EXPORTED_API void addTaskFunc (CtUInt32 nslots, std::string id, F &&func, FArgs &&... fargs)
  Add a task to the service pool with a specified interval and an optional ID.
- EXPORTED_API void removeTask (std::string id)
  Remove a task from the service pool based on its ID.
- EXPORTED_API void startServices ()
  Start the services provided by the service pool.
- EXPORTED_API void shutdownServices ()
  Shutdown the services provided by the service pool.
- EXPORTED_API CtUInt32 getSlotTime ()
  Get slot time.
- EXPORTED_API void setSlotTime (CtUInt32 nslots)
  Set slot time.
- template<typename F , typename... FArgs>
  void **addTaskFunc** (CtUInt32 nslots, std::string id, F &&func, FArgs &&... fargs)

5.19.1 Detailed Description

A service pool for managing and executing tasks at specified intervals using a worker pool.

5.19.2 Constructor & Destructor Documentation

5.19.2.1 CtServicePool()

```
CtServicePool::CtServicePool (
            CtUInt32 nworkers )
```

Constructor for CtServicePool.

Parameters

nworkers	The number of worker threads in the service pool.
slot_time	The time interval for each "slot" in milliseconds. Default is 10 milliseconds.

5.19.3 Member Function Documentation

5.19.3.1 addTask()

```
void CtServicePool::addTask (
            CtUInt32 nslots,
            std::string id,
            CtTask & task )
```

Add a task to the service pool with a specified interval and an optional ID.

Parameters

nslots	The interval in slots for executing the task.
id	An optional ID for the task.
task	The task to be added.

5.19.3.2 addTaskFunc()

```
template<typename F , typename...  FArgs>
EXPORTED_API void CtServicePool::addTaskFunc (
            CtUInt32 nslots,
            std::string id,
            F && func,
            FArgs &&...  fargs )
```

Add a task to the service pool with a specified interval and an optional ID.

Parameters

nslots	The interval in slots for executing the task.
id	An optional ID for the task.
func	The task function to be added.
fargs	The task function's arguments to be added.

5.19.3.3 removeTask()

```
void CtServicePool::removeTask (
            std::string id )
```

Remove a task from the service pool based on its ID.

Parameters

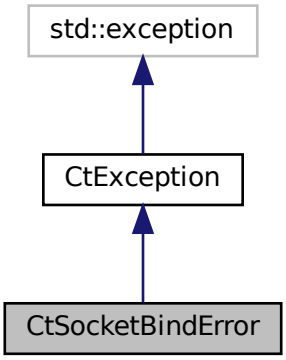
id	The ID of the task to be removed.

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

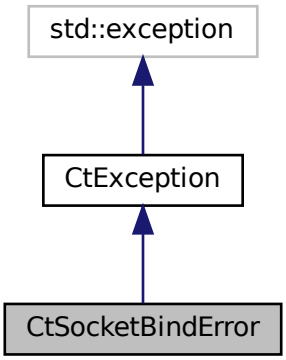
- include/threading/CtServicePool.hpp
- src/threading/CtServicePool.cpp

5.20 CtSocketBindError Class Reference

Inheritance diagram for CtSocketBindError:



Collaboration diagram for CtSocketBindError:



Public Member Functions

- **CtSocketBindError** (const std::string &msg)

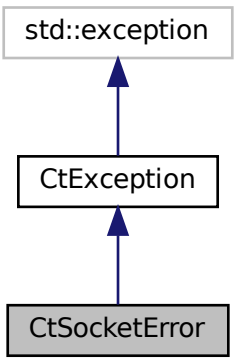
Additional Inherited Members

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

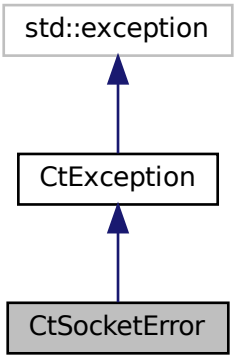
- include/exceptions/CtNetworkExceptions.hpp

5.21 CtSocketError Class Reference

Inheritance diagram for CtSocketError:



Collaboration diagram for CtSocketError:



Public Member Functions

- **CtSocketError** (const std::string &msg)

Additional Inherited Members

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

- include/exceptions/CtNetworkExceptions.hpp

5.22 CtSocketHelpers Class Reference

A class contaning helpers for various sockets utilities.

`#include <CtSocketHelpers.hpp>`

Static Public Member Functions

- static EXPORTED_API void setSocketTimeout (int32_t socketTimeout)
  Set the Socket Timeout object.
- static EXPORTED_API std::vector< std::string > getInterfaces ()
  Get all available interfaces the device.
- static EXPORTED_API std::string interfaceToAddress (const std::string &p_ifName)
  Get address of a specific interface.
- static EXPORTED_API CtUInt32 getAddressAsUInt (const std::string &p_addr)
  Convert address to uin32_t.
- static EXPORTED_API std::string getAddressAsString (CtUInt32 p_addr)
  Convert address to std::string.

Friends

- class **CtSocketUdp**

5.22.1 Detailed Description

A class contaning helpers for various sockets utilities.

5.22.2 Member Function Documentation

5.22.2.1 getAddressAsString()

```
std::string CtSocketHelpers::getAddressAsString (
            CtUInt32 p_addr )  [static]
```

Convert address to std::string.

Parameters

p_addr	The address in form of CtUInt32

5.22.2.2 getAddressAsUInt()

```
CtUInt32 CtSocketHelpers::getAddressAsUInt (
            const std::string & p_addr )  [static]
```

Convert address to uin32_t.

Parameters

p_addr	The address in form of std::string

5.22.2.3 interfaceToAddress()

```
std::string CtSocketHelpers::interfaceToAddress (
            const std::string & p_ifName )  [static]
```

Get address of a specific interface.

Parameters

p_ifName	The name of the interface.

5.22.2.4 setSocketTimeout()

```
void CtSocketHelpers::setSocketTimeout (
            int32_t socketTimeout )  [static]
```

Set the Socket Timeout object.

Parameters

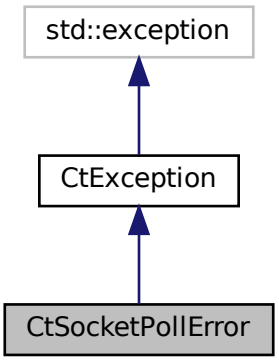
socketTimeout	The target timeout for the poll request.

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

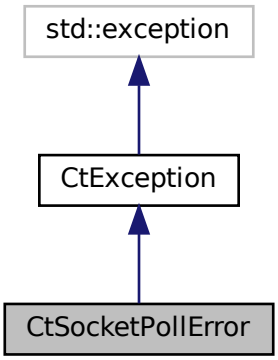
- include/networking/sockets/CtSocketHelpers.hpp
- src/networking/sockets/CtSocketHelpers.cpp

5.23 CtSocketPollError Class Reference

Inheritance diagram for CtSocketPollError:



Collaboration diagram for CtSocketPollError:



Public Member Functions

- **CtSocketPollError** (const std::string &msg)

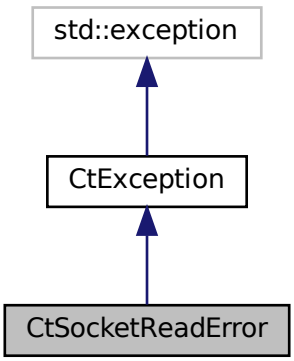
Additional Inherited Members

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

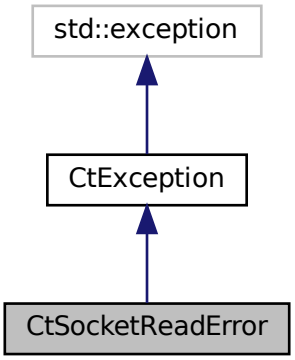
- include/exceptions/CtNetworkExceptions.hpp

5.24 CtSocketReadError Class Reference

Inheritance diagram for CtSocketReadError:



Collaboration diagram for CtSocketReadError:



Public Member Functions

- **CtSocketReadError** (const std::string &msg)

Additional Inherited Members

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

- include/exceptions/CtNetworkExceptions.hpp

5.25 CtSocketUdp Class Reference

A class representing a UDP socket wrapper.

`#include <CtSocketUdp.hpp>`

Public Member Functions

- EXPORTED_API CtSocketUdp ()
  Constructor for CtSocketUdp.
- EXPORTED_API ~CtSocketUdp ()
  Destructor for CtSocketUdp.
- EXPORTED_API void setSub (const std::string &p_interfaceName, uint16_t p_port)
  Set the socket for subscribing.
- EXPORTED_API void setPub (uint16_t p_port, const std::string &p_addr="0.0.0.0")
  Set the socket for publishing.
- EXPORTED_API bool pollRead ()

Check if there is data available to read.

- EXPORTED_API bool pollWrite ()

  Check if data can be written to the fd.
- EXPORTED_API void send (uint8_t *p_data, CtUInt32 p_size)

  Send data over the socket.
- EXPORTED_API void send (CtNetMessage &p_message)

  Send data over the socket.
- EXPORTED_API void receive (uint8_t *p_data, CtUInt32 p_size, CtNetAddress *p_client=nullptr)

  Receive data from the socket.
- EXPORTED_API void receive (CtNetMessage *p_message, CtNetAddress *p_clientAddress=nullptr)

  Receive data from the socket.

5.25.1 Detailed Description

A class representing a UDP socket wrapper.

5.25.2 Member Function Documentation

5.25.2.1 pollRead()

```
bool CtSocketUdp::pollRead ( )
```

Check if there is data available to read.

Returns

True if data is available, false otherwise.

5.25.2.2 pollWrite()

```
bool CtSocketUdp::pollWrite ( )
```

Check if data can be written to the fd.

Returns

True if there is at least one byte available, false otherwise.

5.25.2.3 receive() [1/2]

```
void CtSocketUdp::receive (
            CtNetMessage * p_message,
            CtNetAddress * p_clientAddress = nullptr )
```

Receive data from the socket.

Parameters

p_message	Struct to store the message received.
p_clientAddress	Pointer to a CtNetAddress object to store the client's address (output parameter).

5.25.2.4 receive() [2/2]

```
void CtSocketUdp::receive (
            uint8_t * p_data,
            CtUInt32 p_size,
            CtNetAddress * p_client = nullptr )
```

Receive data from the socket.

Parameters

p_data	Buffer containing the data to sent.
p_size	Size of the buffer.
p_client	Pointer to a CtNetAddress object to store the client's address (output parameter).

5.25.2.5 send() [1/2]

```
void CtSocketUdp::send (
            CtNetMessage & p_message )
```

Send data over the socket.

Parameters

p_message	Struct containing the message to sent.

5.25.2.6 send() [2/2]

```
void CtSocketUdp::send (
            uint8_t * p_data,
            CtUInt32 p_size )
```

Send data over the socket.

Parameters

p_data	Buffer containing the data to sent.
p_size	Size of the buffer.

5.25.2.7 setPub()

```
void CtSocketUdp::setPub (
            uint16_t p_port,
            const std::string & p_addr = "0.0.0.0" )
```

Set the socket for publishing.

Parameters

p_port	The port to send data to.
p_addr	The address to send data to. Default to empty string.

5.25.2.8 setSub()

```
void CtSocketUdp::setSub (
            const std::string & p_interfaceName,
            uint16_t p_port )
```

Set the socket for subscribing.

Parameters

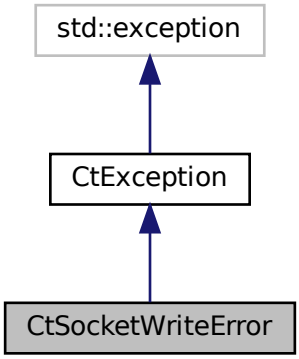
p_interfaceName	The interface name to bind to.
p_port	The port to bind to.

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

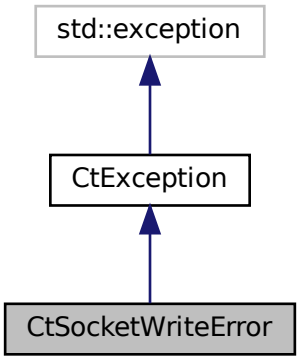
- include/networking/sockets/CtSocketUdp.hpp
- src/networking/sockets/CtSocketUdp.cpp

5.26 CtSocketWriteError Class Reference

Inheritance diagram for CtSocketWriteError:



Collaboration diagram for CtSocketWriteError:



Public Member Functions

- **CtSocketWriteError** (const std::string &msg)

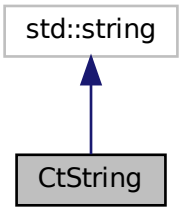
Additional Inherited Members

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

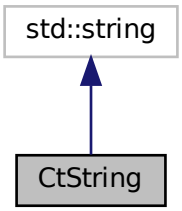
- include/exceptions/CtNetworkExceptions.hpp

5.27 CtString Class Reference

Inheritance diagram for CtString:



Collaboration diagram for CtString:



Public Member Functions

- **CtString** (const std::string &str)
- void **split** (char delimiter, std::vector< CtString > *result) const
- CtString **trim** (const std::string &s) const

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

- include/CtTypes.hpp

5.28 CtTask Class Reference

Represents a task class that encapsulates a callable function (task) and a callback function.

`#include <CtTask.hpp>`

Public Member Functions

- EXPORTED_API CtTask ()

  Default constructor for CtTask. Initializes task and callback with empty lambda functions.
- EXPORTED_API CtTask (const CtTask &other)

  Copy constructor for CtTask. Copies the task and callback from another CtTask object.
- EXPORTED_API ~CtTask ()

  Destructor for CtTask.
- template<typename F , typename... FArgs>
  EXPORTED_API void setTaskFunc (F &&func, FArgs &&... fargs)

  Set the main task function.
- template<typename C , typename... CArgs>
  EXPORTED_API void setCallbackFunc (C &&callback, CArgs &&... cargs)

  Set the callback function.
- EXPORTED_API std::function< void()> getTaskFunc ()

  Get the main task function.
- EXPORTED_API std::function< void()> getCallbackFunc ()

  Get the callback function.
- EXPORTED_API CtTask & operator= (const CtTask &other)

  Assignment operator for CtTask. Copies the task and callback from another CtTask object.
- template<typename F , typename... FArgs>
  void **setTaskFunc** (F &&func, FArgs &&... fargs)
- template<typename C , typename... CArgs>
  void **setCallbackFunc** (C &&callback, CArgs &&... cargs)

5.28.1 Detailed Description

Represents a task class that encapsulates a callable function (task) and a callback function.

5.28.2 Constructor & Destructor Documentation

5.28.2.1 CtTask()

```
CtTask::CtTask (
            const CtTask & other )
```

Copy constructor for CtTask. Copies the task and callback from another CtTask object.

Parameters

other	The CtTask object to copy.

5.28.3 Member Function Documentation

5.28.3.1 getCallbackFunc()

```
std::function< void()> CtTask::getCallbackFunc ( )
```

Get the callback function.

Returns

The callback function.

5.28.3.2 getTaskFunc()

```
std::function< void()> CtTask::getTaskFunc ( )
```

Get the main task function.

Returns

The main task function.

5.28.3.3 operator=()

```
CtTask & CtTask::operator= (
            const CtTask & other )
```

Assignment operator for CtTask. Copies the task and callback from another CtTask object.

Parameters

other	The CtTask object to copy.

Returns

Reference to the current CtTask object.

5.28.3.4 setCallbackFunc()

```
template<typename C , typename...  CArgs>
EXPORTED_API void CtTask::setCallbackFunc (
            C && callback,
            CArgs &&...  cargs )
```

Set the callback function.

Template Parameters

C	Type of the callable function.
CArgs	Types of the arguments for the callable function.

Parameters

callback	The callable function.
cargs	The arguments for the callable function.

5.28.3.5 setTaskFunc()

```
template<typename F , typename...  FArgs>
EXPORTED_API void CtTask::setTaskFunc (
            F && func,
            FArgs &&...  fargs )
```

Set the main task function.

Template Parameters

F	Type of the callable function.
FArgs	Types of the arguments for the callable function.

Parameters

func	The callable function.
fargs	The arguments for the callable function.

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

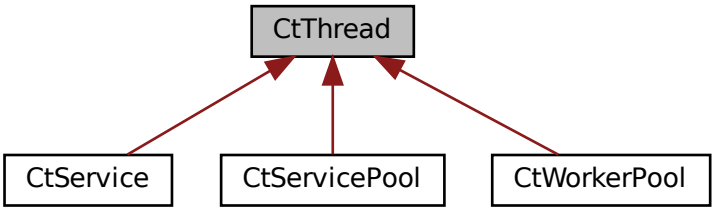
- include/utils/CtTask.hpp
- src/utils/CtTask.cpp

5.29 CtThread Class Reference

A simple C++ thread management class providing basic thread control and sleep functionality.

```
#include <CtThread.hpp>
```

Inheritance diagram for CtThread:



Static Public Member Functions

- static EXPORTED_API void sleepFor (uint64_t time)

  Make the thread sleep for a specified duration in milliseconds.

Protected Member Functions

- EXPORTED_API CtThread ()

  Constructor for CtThread.
- virtual EXPORTED_API ~CtThread ()

  Virtual destructor for CtThread.
- EXPORTED_API bool isRunning ()

  Check if the thread is currently running.
- EXPORTED_API void start ()

  Start the thread.
- EXPORTED_API void stop ()

  Stop the thread.
- EXPORTED_API void join ()

  Join the thread, waiting for it to finish.
- virtual EXPORTED_API void loop ()=0

  Virtual function to be overridden by derived classes. Represents the main functionality of the thread.
- void setRunning (bool running)

  Set the running state of the thread.

5.29.1 Detailed Description

A simple C++ thread management class providing basic thread control and sleep functionality.

5.29.2 Member Function Documentation

5.29.2.1 isRunning()

```
bool CtThread::isRunning ( )  [protected]
```

Check if the thread is currently running.

Returns

True if the thread is running, false otherwise.

5.29.2.2 setRunning()

```
void CtThread::setRunning (
            bool running )  [protected]
```

Set the running state of the thread.

Parameters

running	The running state to set.

5.29.2.3 sleepFor()

```
void CtThread::sleepFor (
            uint64_t time )  [static]
```

Make the thread sleep for a specified duration in milliseconds.

Parameters

time	Duration to sleep in milliseconds.

5.29.2.4 start()

```
void CtThread::start ( )  [protected]
```

Start the thread.

Exceptions

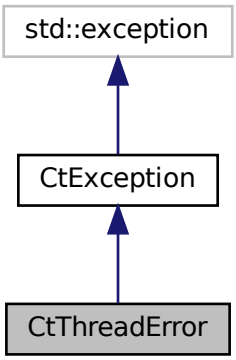
CtThreadError	if the thread is already running.

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

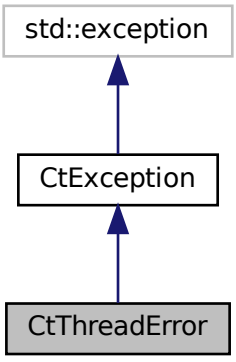
- include/threading/CtThread.hpp
- src/threading/CtThread.cpp

5.30 CtThreadError Class Reference

Inheritance diagram for CtThreadError:



Collaboration diagram for CtThreadError:



Public Member Functions

- **CtThreadError** (const std::string &msg)

Additional Inherited Members

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

- include/exceptions/CtThreadExceptions.hpp

5.31 CtTimer Class Reference

Simple timer utility using std::chrono for high-resolution timing.

`#include <CtTimer.hpp>`

Public Member Functions

- EXPORTED_API CtTimer ()
  Constructor for CtTimer.
- EXPORTED_API ~CtTimer ()
  Destructor for CtTimer.
- EXPORTED_API void tic ()
  Record the current time as a reference point.
- EXPORTED_API uint64_t toc ()
  Measure the elapsed time since the last call to tic().

Static Public Member Functions

- static EXPORTED_API uint64_t current ()
  Get the current time in milliseconds.
- static EXPORTED_API uint64_t millisToNano (uint64_t time)
  Convert time from milliseconds to nanoseconds.

5.31.1 Detailed Description

Simple timer utility using std::chrono for high-resolution timing.

5.31.2 Member Function Documentation

5.31.2.1 current()

```
uint64_t CtTimer::current ( )  [static]
```

Get the current time in milliseconds.

Returns

Current time in milliseconds.

5.31.2.2 millisToNano()

```
static EXPORTED_API uint64_t CtTimer::millisToNano (
            uint64_t time )  [static]
```

Convert time from milliseconds to nanoseconds.

Parameters

time	Time value in milliseconds.

Returns

Time value converted to nanoseconds.

5.31.2.3 toc()

```
uint64_t CtTimer::toc ( )
```

Measure the elapsed time since the last call to tic().

Returns

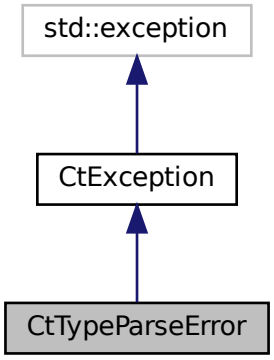
Elapsed time in milliseconds.

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

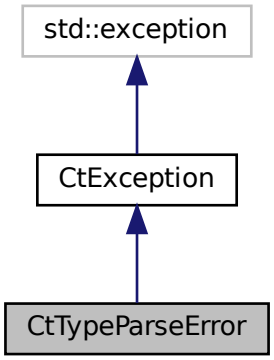
- include/time/CtTimer.hpp
- src/time/CtTimer.cpp

5.32 CtTypeParseError Class Reference

Inheritance diagram for CtTypeParseError:



Collaboration diagram for CtTypeParseError:



Public Member Functions

- **CtTypeParseError** (const std::string &msg)

Additional Inherited Members

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

- include/exceptions/CtTypeExceptions.hpp

5.33 CtWorker Class Reference

Represents a worker thread that can execute tasks asynchronously.

`#include <CtWorker.hpp>`

Public Member Functions

- EXPORTED_API CtWorker ()
  Constructor for CtWorker.
- EXPORTED_API ~CtWorker ()
  Destructor for CtWorker.
- EXPORTED_API bool isRunning ()
  Returns true if the worker is currently running.
- EXPORTED_API void runTask ()
  Run the task assigned to the worker.
- EXPORTED_API void joinTask ()

Join the worker's thread, waiting for the task to complete.

- EXPORTED_API void setTask (CtTask &task, std::function< void()> callback=[ ]{})

  Set a task for the worker to execute.

- template<typename F , typename... FArgs>
  EXPORTED_API void setTaskFunc (F &&func, FArgs &&... fargs)

  Set a task function for the worker to execute.

- template<typename F , typename... FArgs>
  void **setTaskFunc** (F &&func, FArgs &&... fargs)

5.33.1 Detailed Description

Represents a worker thread that can execute tasks asynchronously.

5.33.2 Member Function Documentation

5.33.2.1 isRunning()

```
bool CtWorker::isRunning ( )
```

Returns true if the worker is currently running.

Returns

EXPORTED_API Worker status.

5.33.2.2 setTask()

```
void CtWorker::setTask (
            CtTask & task,
            std::function< void()> callback = []{} )
```

Set a task for the worker to execute.

Parameters

task	The task to be executed by the worker.
callback	The callback function to be executed after the task is completed. Default is an empty lambda function.

5.33.2.3 setTaskFunc()

```
template<typename F , typename...  FArgs>
```

```
EXPORTED_API void CtWorker::setTaskFunc (
            F && func,
            FArgs &&...  fargs )
```

Set a task function for the worker to execute.

Parameters

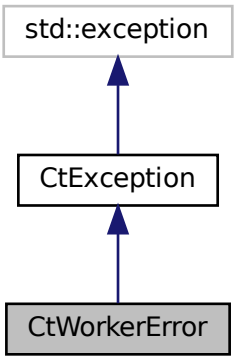
func	The task function to be executed by the worker.
fargs	The arguments of the executed task function.

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

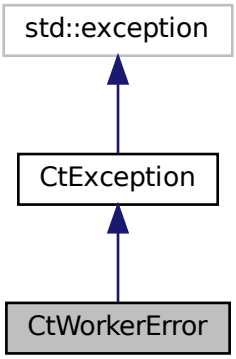
- include/threading/CtWorker.hpp
- src/threading/CtWorker.cpp

5.34 CtWorkerError Class Reference

Inheritance diagram for CtWorkerError:



Collaboration diagram for CtWorkerError:



Public Member Functions

- **CtWorkerError** (const std::string &msg)

Additional Inherited Members

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

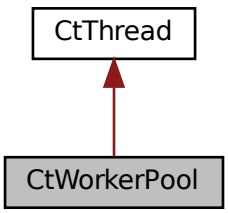
- include/exceptions/CtThreadExceptions.hpp

5.35 CtWorkerPool Class Reference

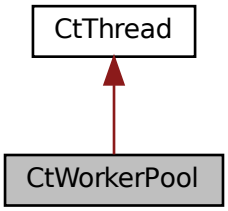
Manages a pool of worker threads for executing tasks concurrently.

`#include <CtWorkerPool.hpp>`

Inheritance diagram for CtWorkerPool:



Collaboration diagram for CtWorkerPool:



Public Member Functions

- EXPORTED_API CtWorkerPool (CtUInt32 nworkers)

  Constructor for CtWorkerPool.
- EXPORTED_API ~CtWorkerPool ()

  Destructor for CtWorkerPool.
- EXPORTED_API void addTask (CtTask &task)

  Add a task to the worker pool.
- template<typename F , typename... FArgs>
  EXPORTED_API void addTask (F &&func, FArgs &&... fargs)

  Add a task function to the worker pool.
- EXPORTED_API void join ()

  Wait for all worker threads to finish their tasks.
- template<typename F , typename... FArgs>
  void **addTask** (F &&func, FArgs &&... fargs)

5.35.1 Detailed Description

Manages a pool of worker threads for executing tasks concurrently.

5.35.2 Constructor & Destructor Documentation

5.35.2.1 CtWorkerPool()

```
CtWorkerPool::CtWorkerPool (
            CtUInt32 nworkers )
```

Constructor for CtWorkerPool.

Parameters

nworkers	The number of worker threads in the pool.

5.35.3 Member Function Documentation

5.35.3.1 addTask() [1/2]

```
void CtWorkerPool::addTask (
            CtTask & task )
```

Add a task to the worker pool.

Parameters

task	The task to be added to the pool.

5.35.3.2 addTask() [2/2]

```
template<typename F , typename...  FArgs>
EXPORTED_API void CtWorkerPool::addTask (
            F && func,
            FArgs &&...  fargs )
```

Add a task function to the worker pool.

Parameters

func	The task function to be added to the pool.
fargs	The arguments of the task function.

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

- include/threading/CtWorkerPool.hpp
- src/threading/CtWorkerPool.cpp

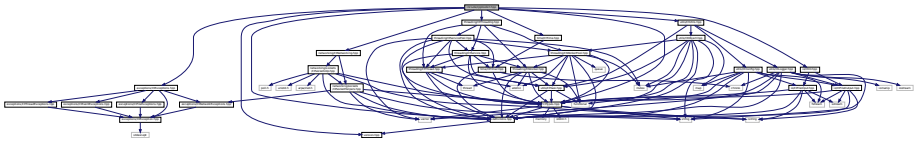
Chapter 6

File Documentation

6.1 include/cpptoolkit.hpp File Reference

```
#include "version.hpp"
#include "definitions.hpp"
#include "CtTypes.hpp"
#include "exceptions/CtExceptions.hpp"
#include "io/CtIO.hpp"
#include "networking/CtNetworking.hpp"
#include "threading/CtThreading.hpp"
#include "time/CtTime.hpp"
#include "utils/CtUtils.hpp"
```

Include dependency graph for cpptoolkit.hpp:



6.1.1 Detailed Description

Date

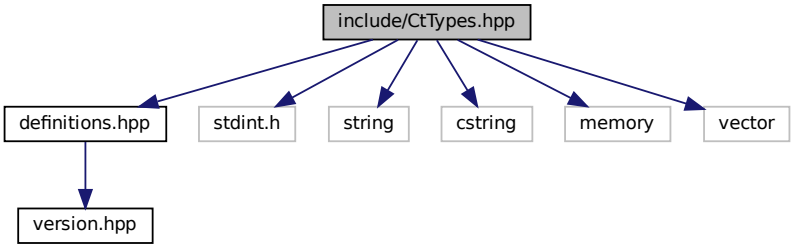
10-01-2025

6.2 include/CtTypes.hpp File Reference

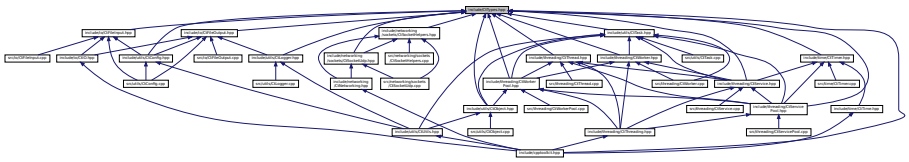
```
#include "definitions.hpp"
#include <stdint.h>
#include <string>
#include <cstring>
#include <memory>
```

```
#include <vector>
```

Include dependency graph for CtTypes.hpp:



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



Classes

- class CtString
- struct _CtNetAddress

  Struct describing a network address.
- struct _CtNetMessage

  Struct describing data sent over network.
- class CtRawData

  Struct describing raw data.

Macros

- #define **CtUInt8** uint8_t
- #define **CtUInt16** uint16_t
- #define **CtUInt32** uint32_t
- #define **CtUInt64** uint64_t
- #define **CtInt8** int8_t
- #define **CtInt16** int16_t
- #define **CtInt32** int32_t
- #define **CtInt64** int64_t
- #define **CtChar** char
- #define **CTNET_BUFFER_SIZE** 2048

Typedefs

- typedef struct _CtNetAddress CtNetAddress

  Struct describing a network address.
- typedef struct _CtNetMessage CtNetMessage

  Struct describing data sent over network.

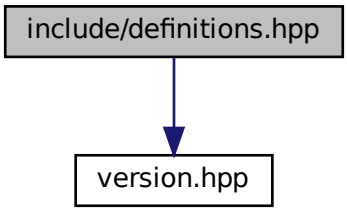
6.2.1 Detailed Description

Date

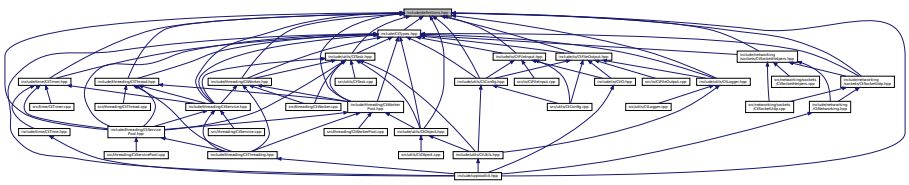
21-01-2024

6.3 include/definitions.hpp File Reference

`#include "version.hpp"`
Include dependency graph for definitions.hpp:



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



Macros

- #define **EXPORTED_API** __attribute__((visibility("default")))

6.3.1 Detailed Description

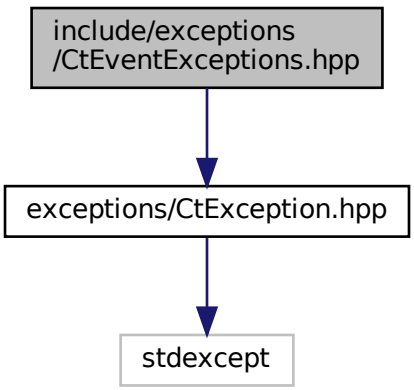
Date

18-01-2024

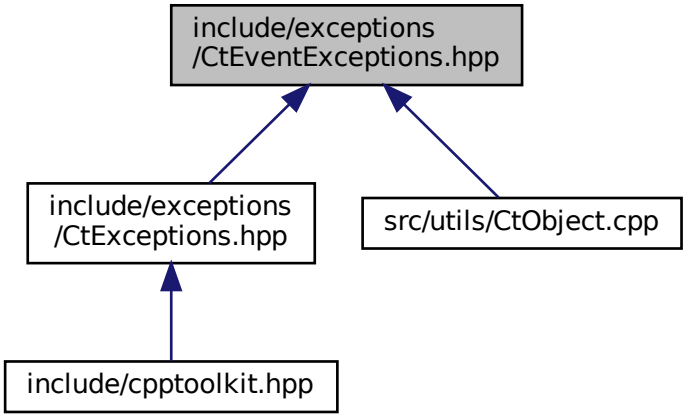
6.4 include/exceptions/CtEventExceptions.hpp File Reference

```
#include "exceptions/CtException.hpp"
```

Include dependency graph for CtEventExceptions.hpp:



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



Classes

- class CtEventNotExistsError
- class CtEventAlreadyExistsError

6.4.1 Detailed Description

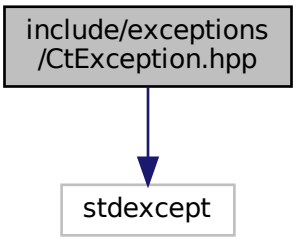
Date

02-02-2024

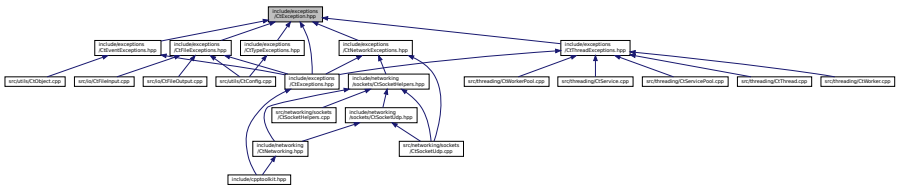
6.5 include/exceptions/CtException.hpp File Reference

```
#include <stdexcept>
```

Include dependency graph for CtException.hpp:



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



Classes

- class CtException

6.5.1 Detailed Description

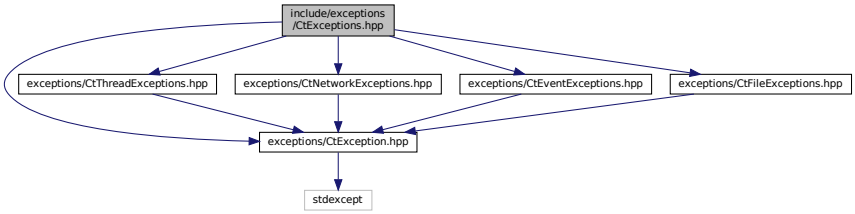
Date

18-01-2024

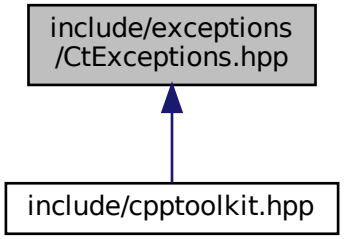
6.6 include/exceptions/CtExceptions.hpp File Reference

```
#include "exceptions/CtException.hpp"
#include "exceptions/CtThreadExceptions.hpp"
#include "exceptions/CtNetworkExceptions.hpp"
#include "exceptions/CtEventExceptions.hpp"
#include "exceptions/CtFileExceptions.hpp"
```

Include dependency graph for CtExceptions.hpp:



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



6.6.1 Detailed Description

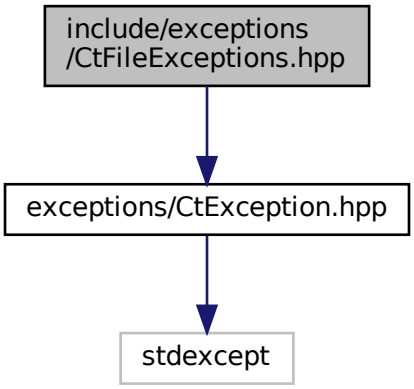
Date

18-01-2024

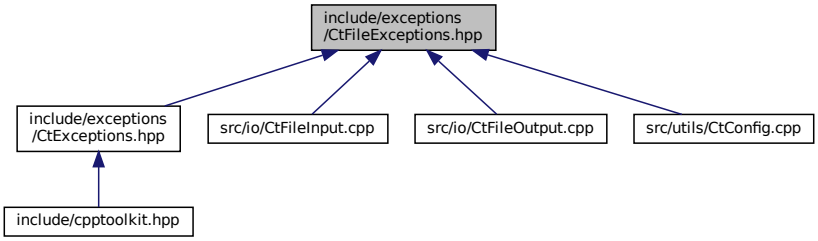
6.7 include/exceptions/CtFileExceptions.hpp File Reference

```
#include "exceptions/CtException.hpp"
```

Include dependency graph for CtFileExceptions.hpp:



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



Classes

- class CtFileReadError
- class CtFileWriteError
- class CtFileParseError

6.7.1 Detailed Description

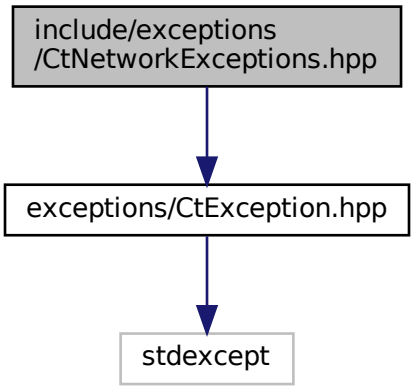
Date

10-03-2024

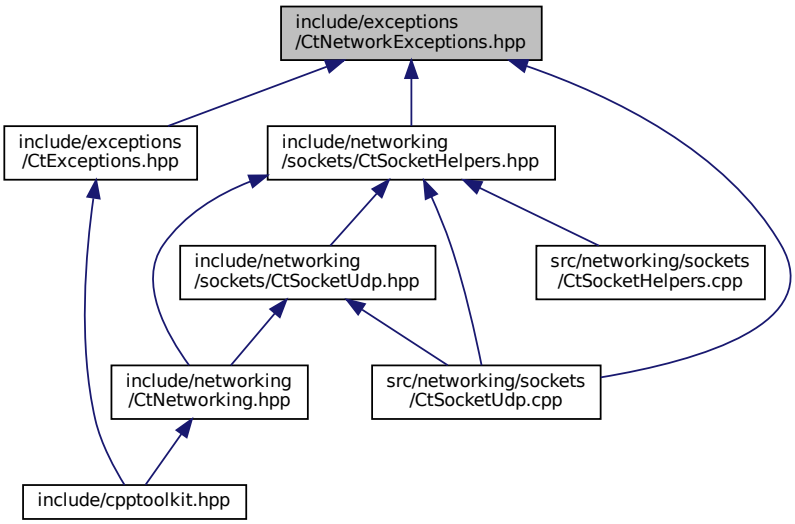
6.8 include/exceptions/CtNetworkExceptions.hpp File Reference

```
#include "exceptions/CtException.hpp"
```

Include dependency graph for CtNetworkExceptions.hpp:



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



Classes

- class CtSocketError
- class CtSocketBindError
- class CtSocketPollError
- class CtSocketReadError
- class CtSocketWriteError

6.8.1 Detailed Description

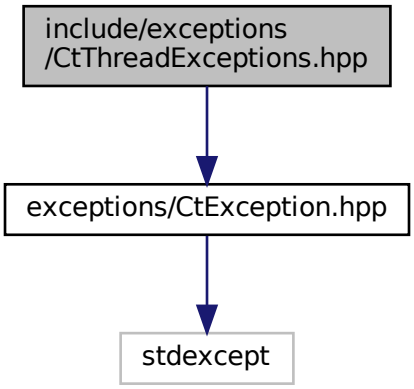
Date

18-01-2024

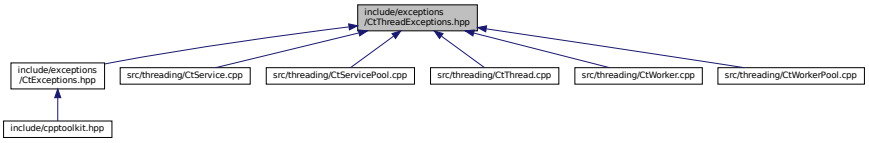
6.9 include/exceptions/CtThreadExceptions.hpp File Reference

```
#include "exceptions/CtException.hpp"
```

Include dependency graph for CtThreadExceptions.hpp:



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



Classes

- class CtThreadError
- class CtServiceError
- class CtWorkerError

6.9.1 Detailed Description

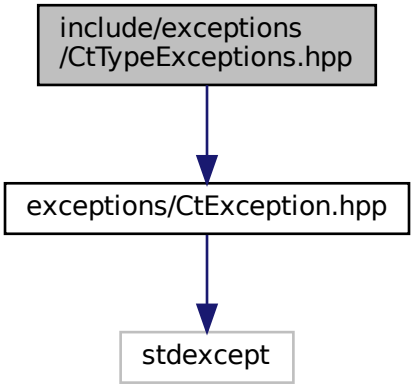
Date

18-01-2024

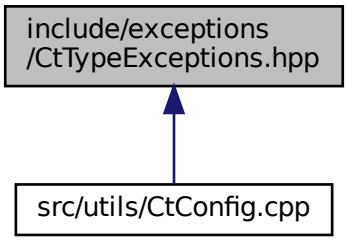
6.10 include/exceptions/CtTypeExceptions.hpp File Reference

```
#include "exceptions/CtException.hpp"
```

Include dependency graph for CtTypeExceptions.hpp:



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



Classes

- class CtTypeParseError
- class CtKeyNotFoundError
- class CtDataTypeInvalid

6.10.1 Detailed Description

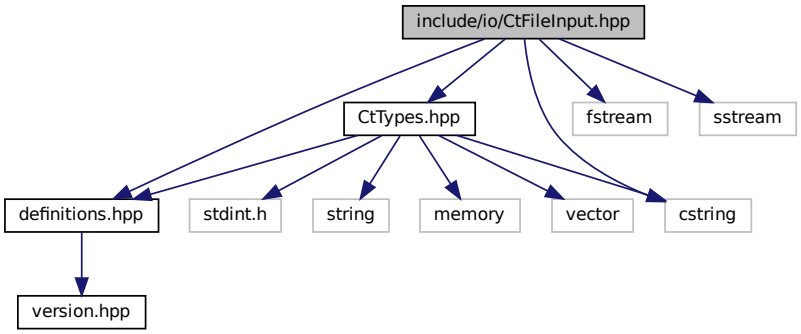
Date

10-03-2024

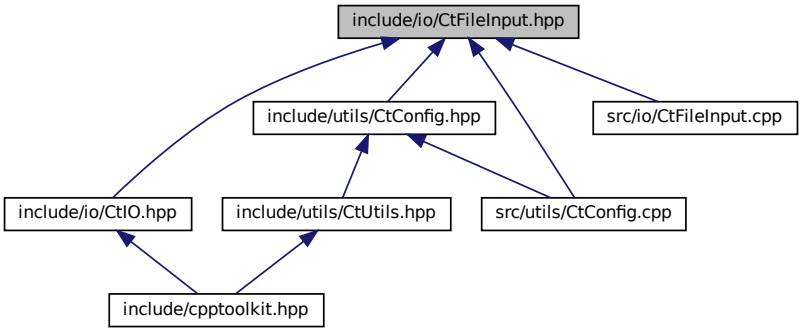
6.11 include/io/CtFileInput.hpp File Reference

```
#include "definitions.hpp"
#include "CtTypes.hpp"
#include <fstream>
#include <sstream>
#include <cstring>
```

Include dependency graph for CtFileInput.hpp:



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



Classes

- class CtFileInput

6.11.1 Detailed Description

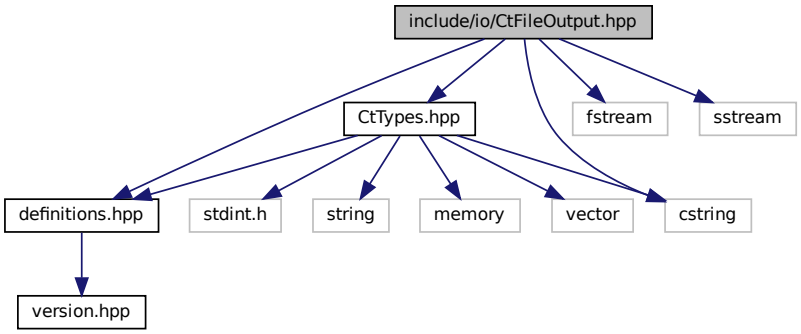
Date

08-03-2024

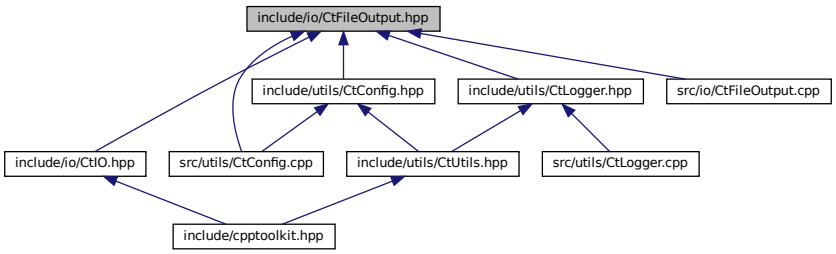
6.12 include/io/CtFileOutput.hpp File Reference

```
#include "definitions.hpp"
#include "CtTypes.hpp"
#include <fstream>
#include <sstream>
#include <cstring>
```

Include dependency graph for CtFileOutput.hpp:



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



Classes

- class CtFileOutput

6.12.1 Detailed Description

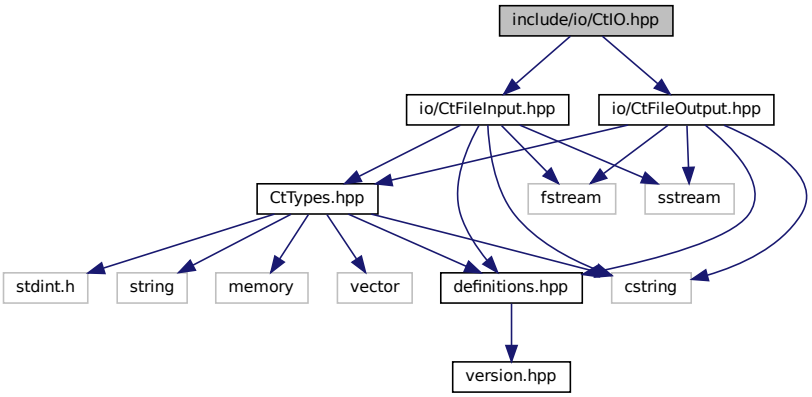
Date

09-03-2024

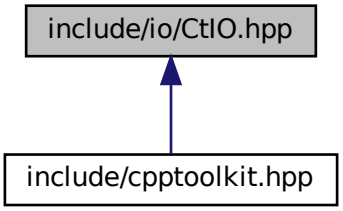
6.13 include/io/CtIO.hpp File Reference

```
#include "io/CtFileOutput.hpp"
#include "io/CtFileInput.hpp"
```

Include dependency graph for CtIO.hpp:



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



6.13.1 Detailed Description

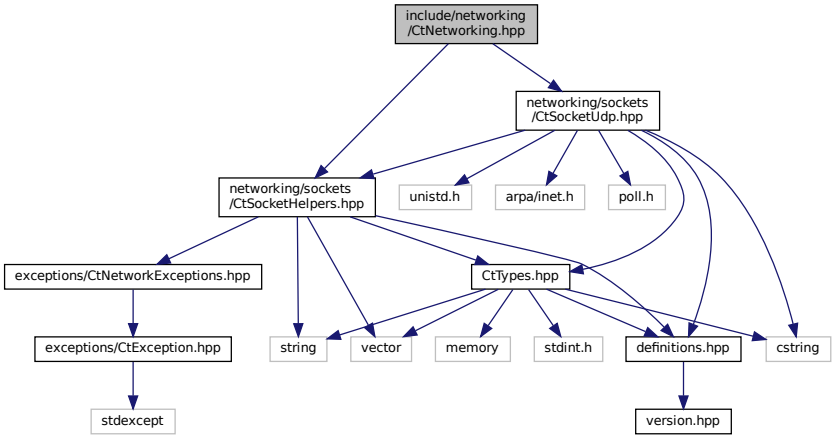
Date

19-03-2024

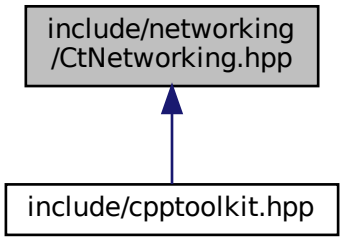
6.14 include/networking/CtNetworking.hpp File Reference

```
#include "networking/sockets/CtSocketHelpers.hpp"
#include "networking/sockets/CtSocketUdp.hpp"
```

Include dependency graph for CtNetworking.hpp:



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



6.14.1 Detailed Description

Date

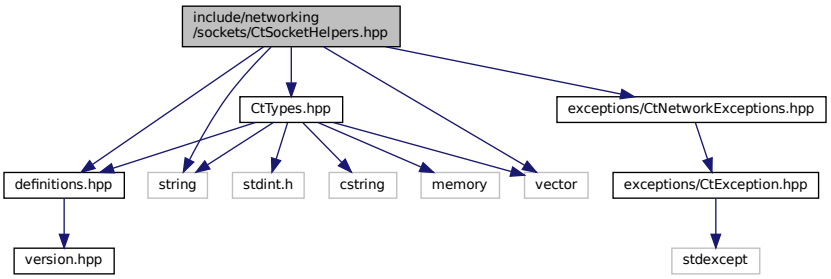
18-01-2024

6.15 include/networking/sockets/CtSocketHelpers.hpp File Reference

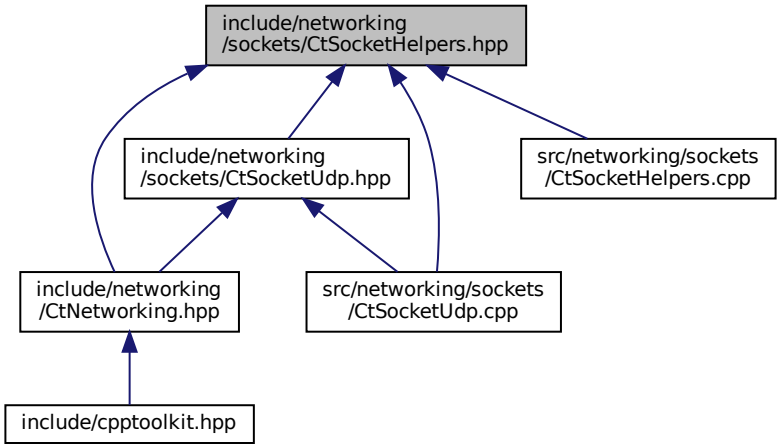
```
#include "definitions.hpp"
#include "CtTypes.hpp"
#include "exceptions/CtNetworkExceptions.hpp"
#include <string>
```

```
#include <vector>
```

Include dependency graph for CtSocketHelpers.hpp:



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



Classes

- class CtSocketHelpers

  A class contaning helpers for various sockets utilities.

6.15.1 Detailed Description

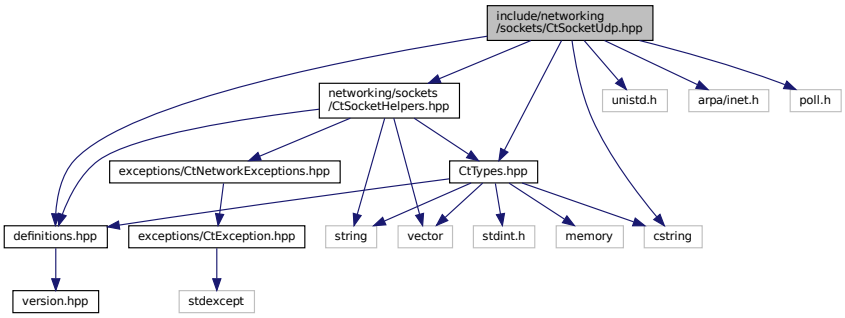
Date

21-01-2024

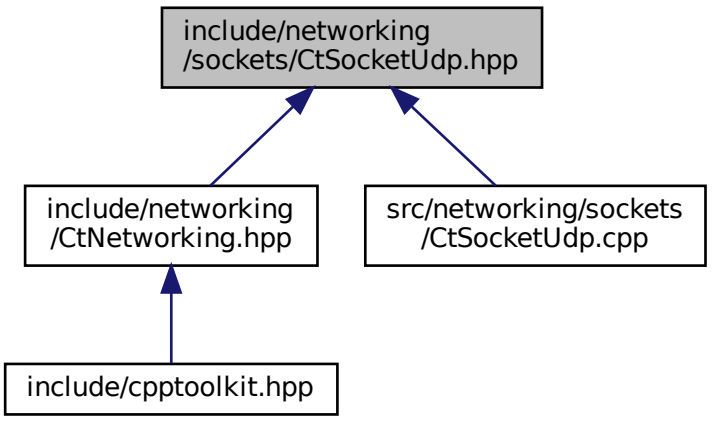
6.16 include/networking/sockets/CtSocketUdp.hpp File Reference

```
#include "definitions.hpp"
#include "CtTypes.hpp"
#include "networking/sockets/CtSocketHelpers.hpp"
#include <cstring>
#include <unistd.h>
#include <arpa/inet.h>
#include <poll.h>
```

Include dependency graph for CtSocketUdp.hpp:



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



Classes

- class CtSocketUdp

  A class representing a UDP socket wrapper.

6.16.1 Detailed Description

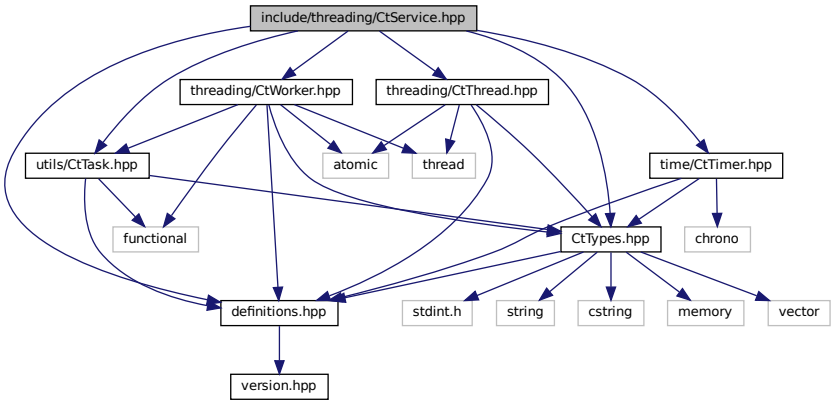
Date

18-01-2024

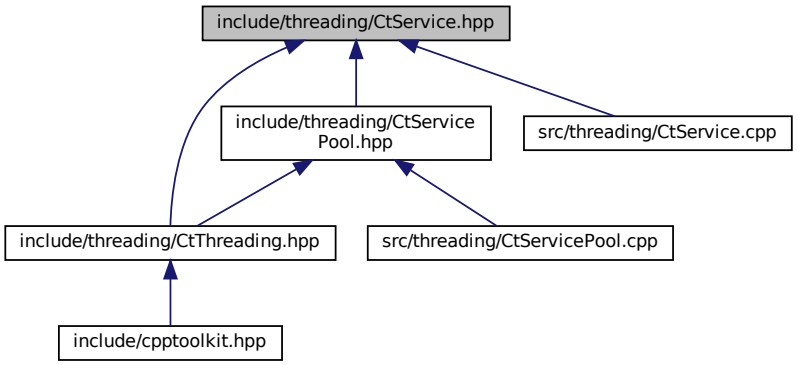
6.17 include/threading/CtService.hpp File Reference

```
#include "definitions.hpp"
#include "CtTypes.hpp"
#include "threading/CtThread.hpp"
#include "threading/CtWorker.hpp"
#include "utils/CtTask.hpp"
#include "time/CtTimer.hpp"
```

Include dependency graph for CtService.hpp:



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



Classes

- class CtService

  A class representing a service that runs a given task at regular intervals using a worker thread.

6.17.1 Detailed Description

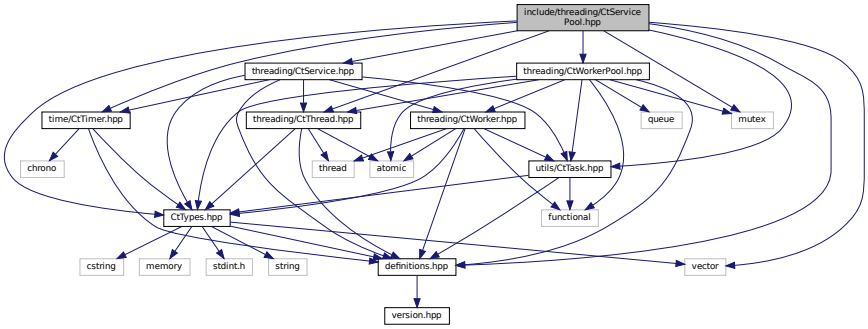
Date

18-01-2024

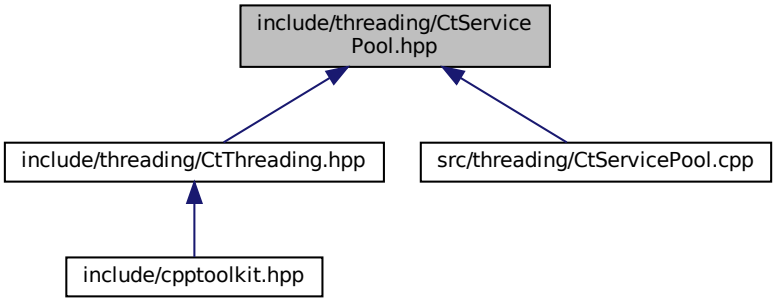
6.18 include/threading/CtServicePool.hpp File Reference

```
#include "definitions.hpp"
#include "CtTypes.hpp"
#include "threading/CtService.hpp"
#include "threading/CtWorkerPool.hpp"
#include "threading/CtThread.hpp"
#include "time/CtTimer.hpp"
#include "utils/CtTask.hpp"
#include <vector>
#include <mutex>
```

Include dependency graph for CtServicePool.hpp:



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



Classes

- class CtServicePool

  A service pool for managing and executing tasks at specified intervals using a worker pool.

6.18.1 Detailed Description

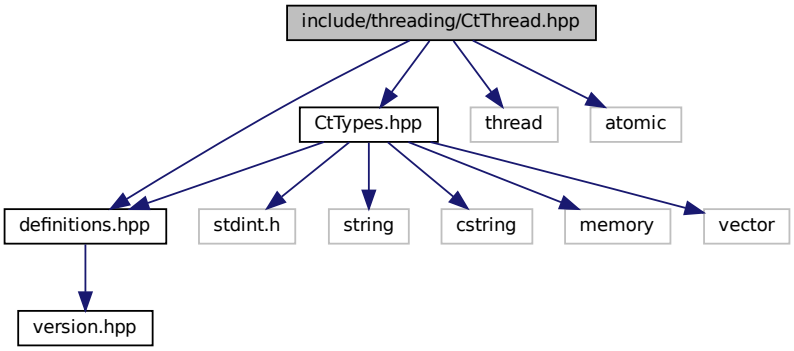
Date

18-01-2024

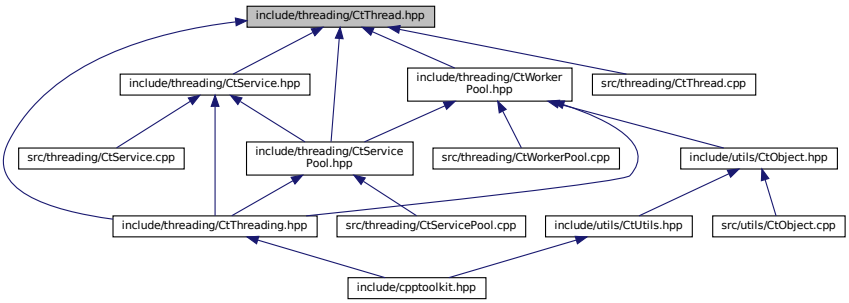
6.19 include/threading/CtThread.hpp File Reference

```
#include "definitions.hpp"
#include "CtTypes.hpp"
#include <thread>
#include <atomic>
```

Include dependency graph for CtThread.hpp:



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



Classes

- class CtThread

  A simple C++ thread management class providing basic thread control and sleep functionality.

6.19.1 Detailed Description

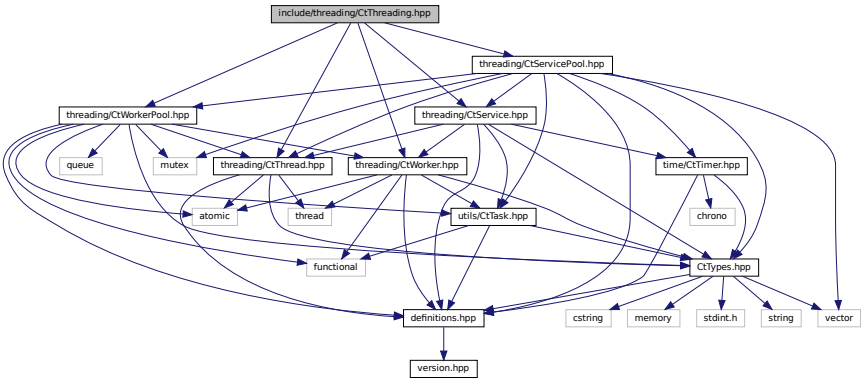
Date

18-01-2024

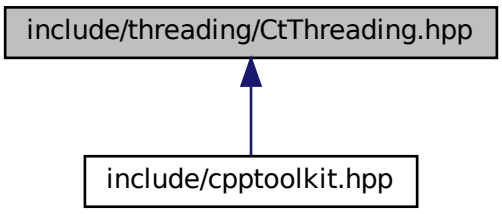
6.20 include/threading/CtThreading.hpp File Reference

```
#include "threading/CtService.hpp"
#include "threading/CtServicePool.hpp"
#include "threading/CtThread.hpp"
#include "threading/CtWorker.hpp"
#include "threading/CtWorkerPool.hpp"
```

Include dependency graph for CtThreading.hpp:



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



6.20.1 Detailed Description

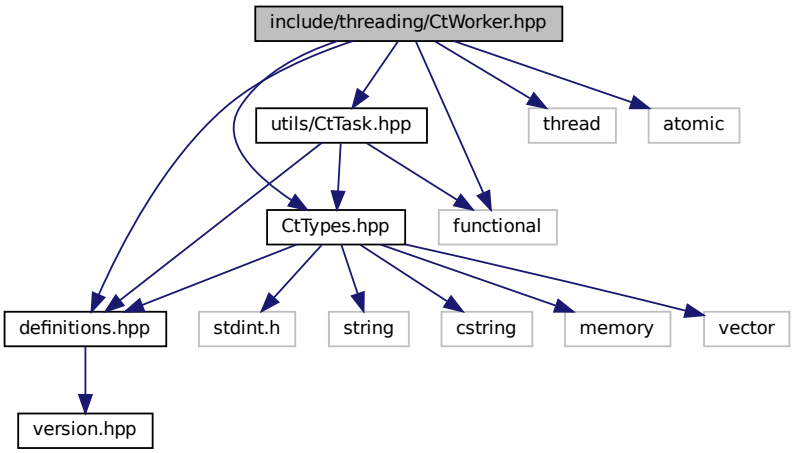
Date

18-01-2024

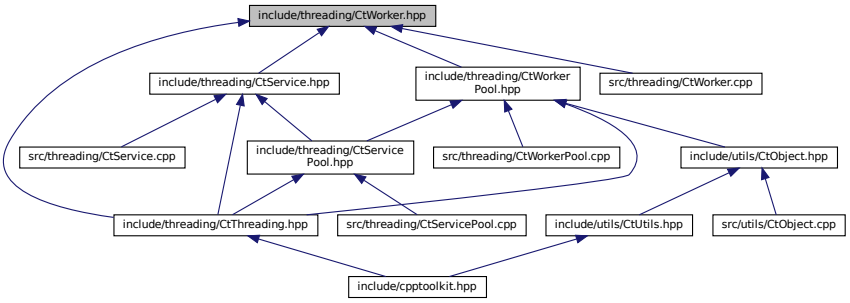
6.21 include/threading/CtWorker.hpp File Reference

```
#include "definitions.hpp"
#include "CtTypes.hpp"
#include "utils/CtTask.hpp"
#include <thread>
#include <atomic>
#include <functional>
```

Include dependency graph for CtWorker.hpp:



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



Classes

- class CtWorker

  Represents a worker thread that can execute tasks asynchronously.

6.21.1 Detailed Description

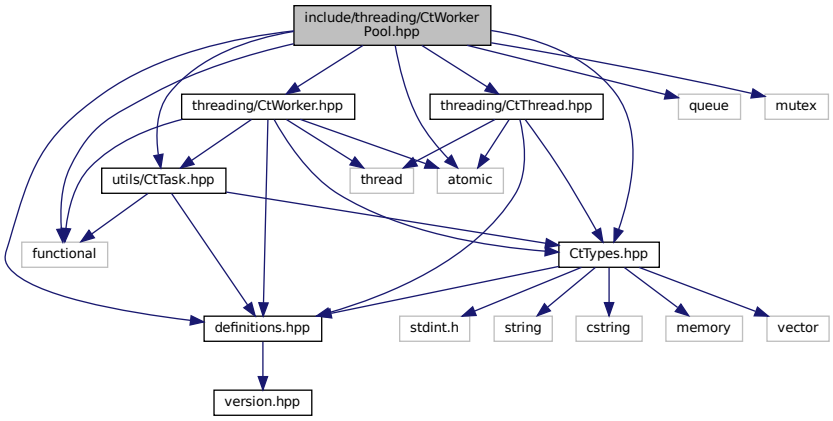
Date

18-01-2024

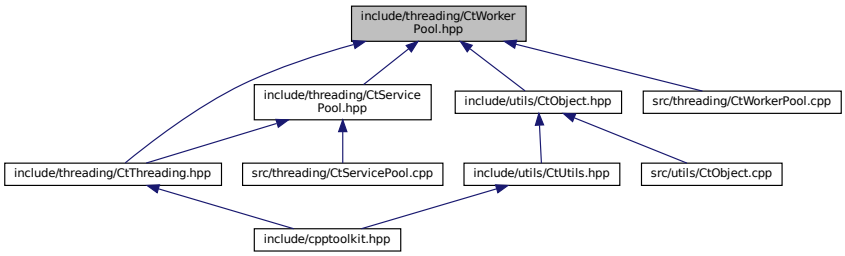
6.22 include/threading/CtWorkerPool.hpp File Reference

```
#include "definitions.hpp"
#include "CtTypes.hpp"
#include "threading/CtWorker.hpp"
#include "threading/CtThread.hpp"
#include "utils/CtTask.hpp"
#include <queue>
#include <atomic>
#include <mutex>
#include <functional>
```

Include dependency graph for CtWorkerPool.hpp:



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



Classes

- class CtWorkerPool

  Manages a pool of worker threads for executing tasks concurrently.

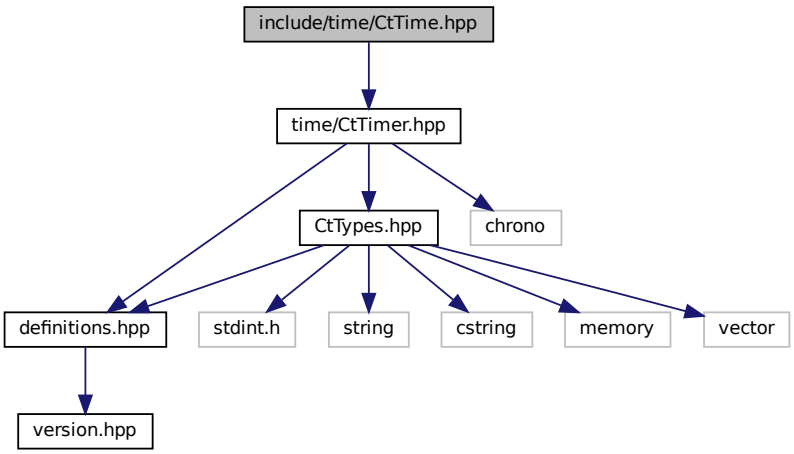
6.22.1 Detailed Description

Date

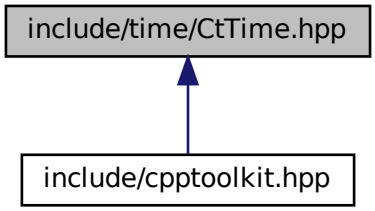
18-01-2024

6.23 include/time/CtTime.hpp File Reference

`#include "time/CtTimer.hpp"`
Include dependency graph for CtTime.hpp:



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



6.23.1 Detailed Description

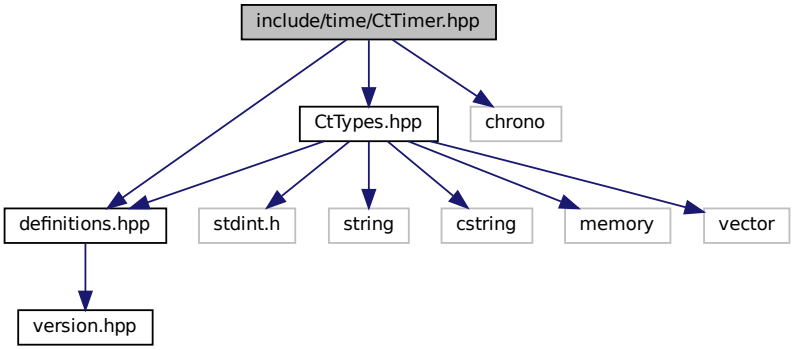
Date

18-01-2024

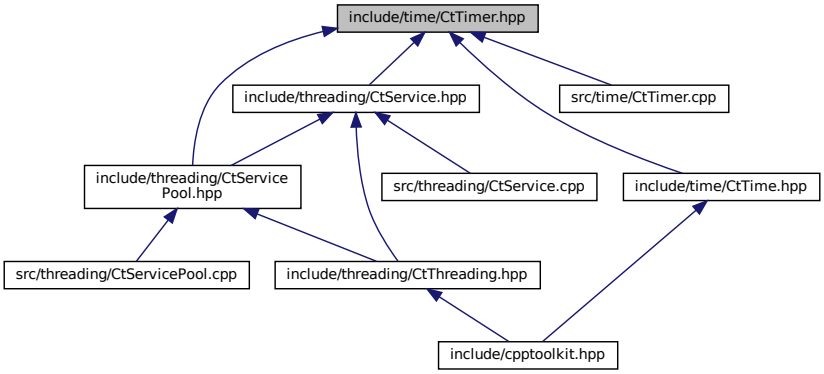
6.24 include/time/CtTimer.hpp File Reference

```
#include "definitions.hpp"
#include "CtTypes.hpp"
#include <chrono>
```

Include dependency graph for CtTimer.hpp:



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



Classes

- class CtTimer

  Simple timer utility using std::chrono for high-resolution timing.

6.24.1 Detailed Description

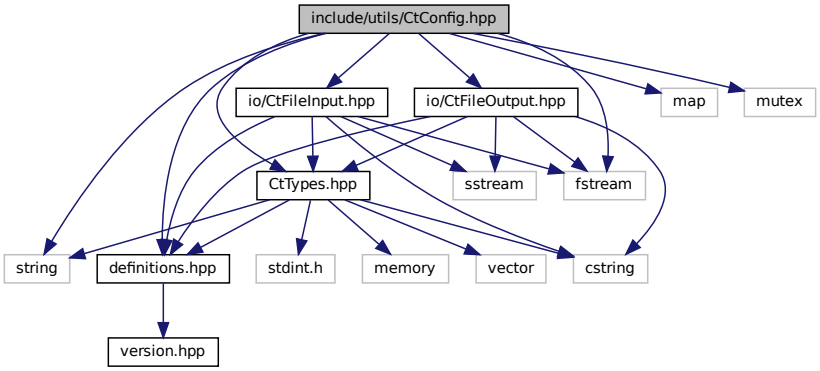
Date

18-01-2024

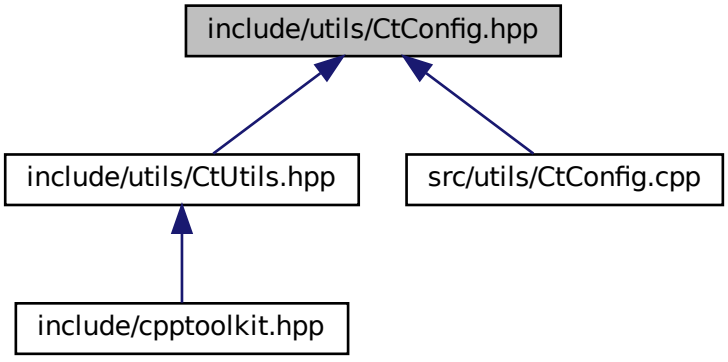
6.25 include/utils/CtConfig.hpp File Reference

```
#include "definitions.hpp"
#include "CtTypes.hpp"
#include "io/CtFileOutput.hpp"
#include "io/CtFileInput.hpp"
#include <fstream>
#include <string>
#include <map>
#include <mutex>
```

Include dependency graph for CtConfig.hpp:



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



Classes

- class CtConfig

  A configuration file parser class for extracting various data types from configuration values.

6.25.1 Detailed Description

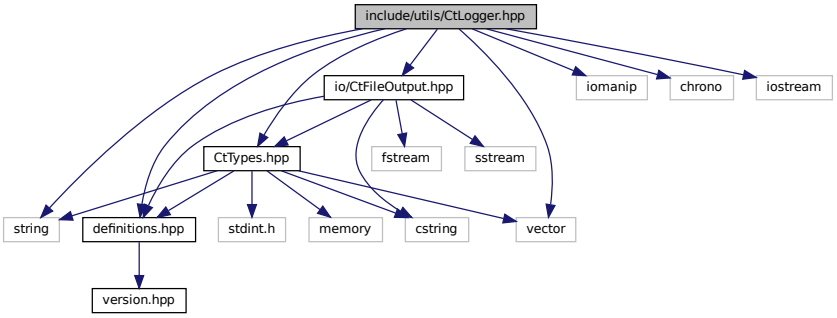
Date

10-03-2024

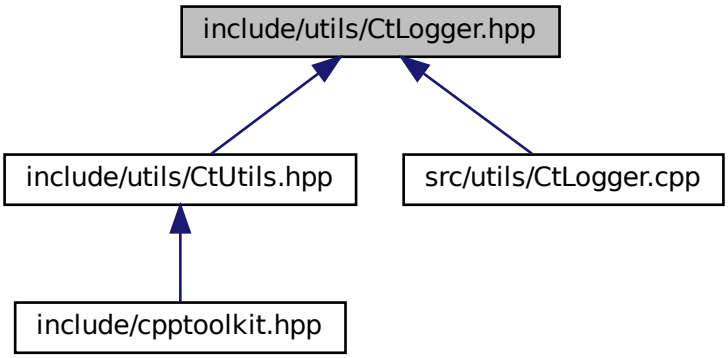
6.26 include/utils/CtLogger.hpp File Reference

```
#include "definitions.hpp"
#include "CtTypes.hpp"
#include "io/CtFileOutput.hpp"
#include <string>
#include <iomanip>
#include <chrono>
#include <vector>
#include <iostream>
```

Include dependency graph for CtLogger.hpp:



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



Classes

- class CtLogger

  A simple logger with log levels and timestamp.

6.26.1 Detailed Description

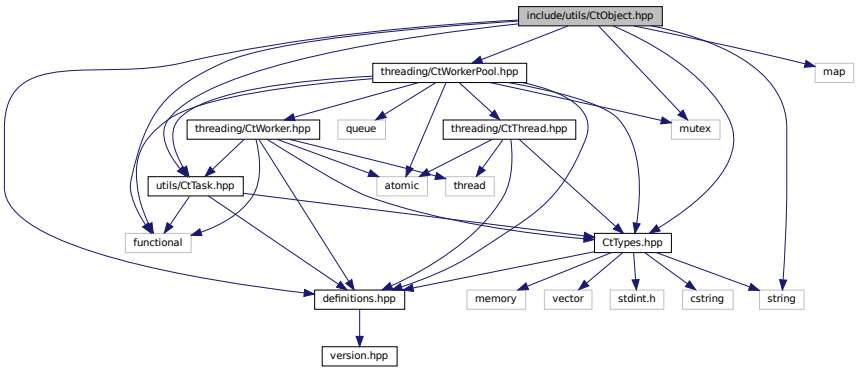
Date

10-03-2024

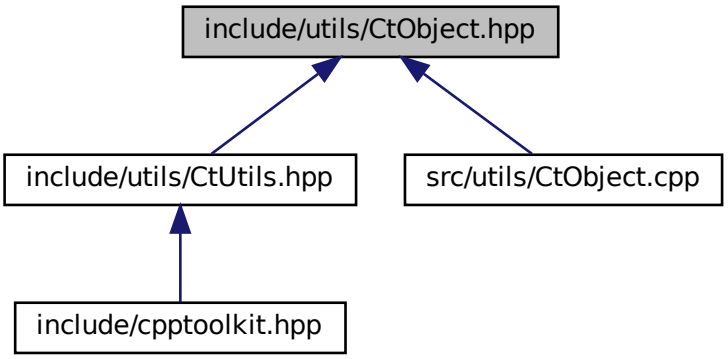
6.27 include/utils/CtObject.hpp File Reference

```
#include "definitions.hpp"
#include "CtTypes.hpp"
#include "utils/CtTask.hpp"
#include "threading/CtWorkerPool.hpp"
#include <string>
#include <map>
#include <mutex>
#include <functional>
```

Include dependency graph for CtObject.hpp:



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



Classes

- class CtObject

  This abstract class can be used as a base class for objects that can trigger events.

6.27.1 Detailed Description

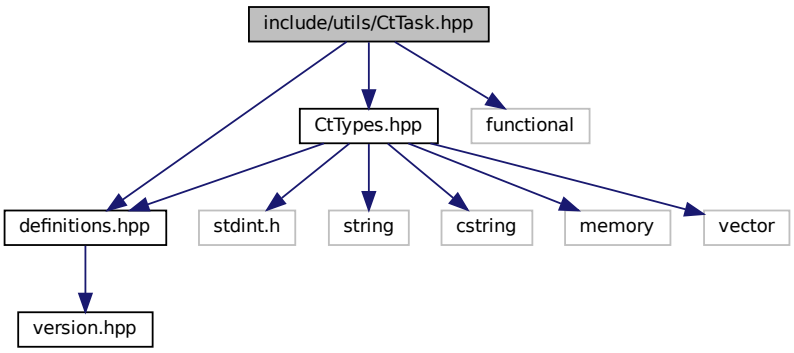
Date

02-02-2024

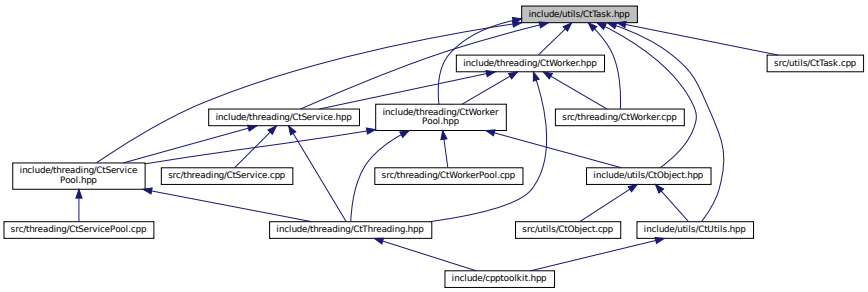
6.28 include/utils/CtTask.hpp File Reference

```
#include "definitions.hpp"
#include "CtTypes.hpp"
#include <functional>
```

Include dependency graph for CtTask.hpp:



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



Classes

- class CtTask

  Represents a task class that encapsulates a callable function (task) and a callback function.

6.28.1 Detailed Description

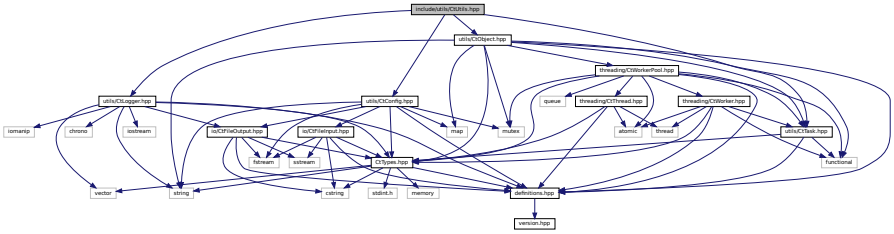
Date

18-01-2024

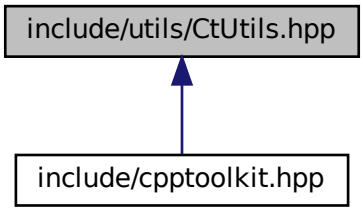
6.29 include/utils/CtUtils.hpp File Reference

```
#include "utils/CtTask.hpp"
#include "utils/CtObject.hpp"
#include "utils/CtLogger.hpp"
#include "utils/CtConfig.hpp"
```

Include dependency graph for CtUtils.hpp:



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



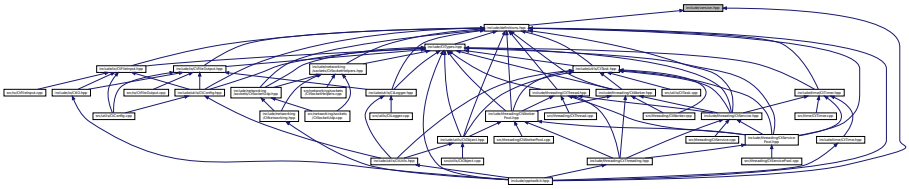
6.29.1 Detailed Description

Date

18-01-2024

6.30 include/version.hpp File Reference

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



Macros

- #define **CPPTOOLKIT_VERSION_MAJOR** 0
- #define **CPPTOOLKIT_VERSION_MINOR** 1
- #define **CPPTOOLKIT_VERSION_PATCH** 0
- #define **CPPTOOLKIT_VERSION** (CPPTOOLKIT_VERSION_MAJOR ## "." ## CPPTOOLKIT_VERSION↩ _MINOR ## "." ## CPPTOOLKIT_VERSION_PATCH)

6.30.1 Detailed Description

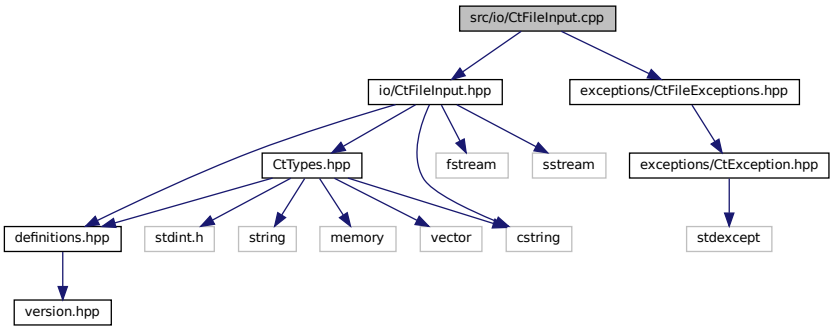
Date

18-01-2024

6.31 src/io/CtFileInput.cpp File Reference

```
#include "io/CtFileInput.hpp"
#include "exceptions/CtFileExceptions.hpp"
```

Include dependency graph for CtFileInput.cpp:



6.31.1 Detailed Description

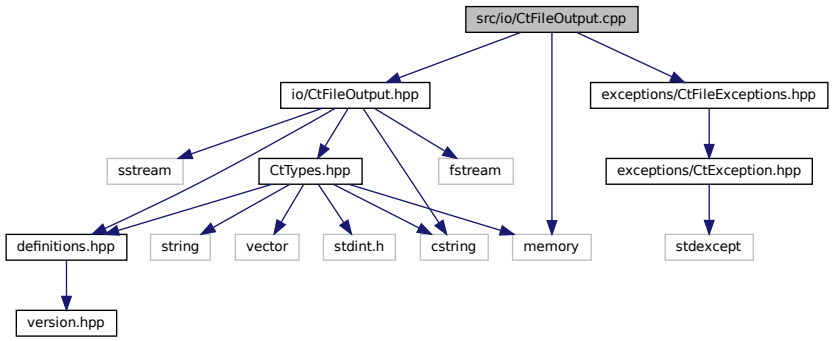
Date

08-03-2024

6.32 src/io/CtFileOutput.cpp File Reference

```
#include "io/CtFileOutput.hpp"
#include "exceptions/CtFileExceptions.hpp"
#include <memory>
```

Include dependency graph for CtFileOutput.cpp:



6.32.1 Detailed Description

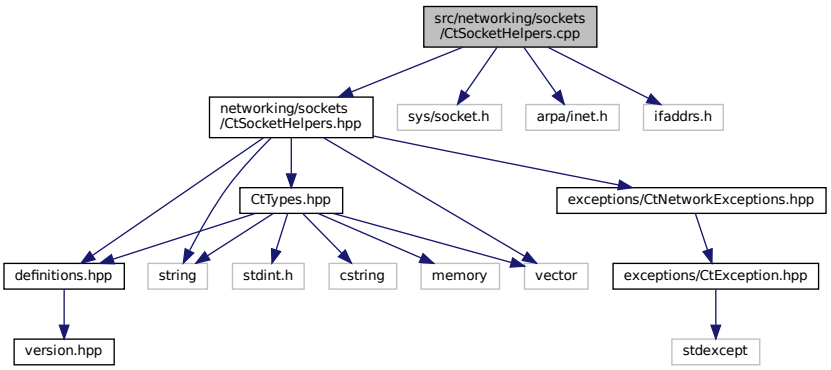
Date

09-03-2024

6.33 src/networking/sockets/CtSocketHelpers.cpp File Reference

```
#include "networking/sockets/CtSocketHelpers.hpp"
#include <sys/socket.h>
#include <arpa/inet.h>
#include <ifaddrs.h>
```

Include dependency graph for CtSocketHelpers.cpp:



6.33.1 Detailed Description

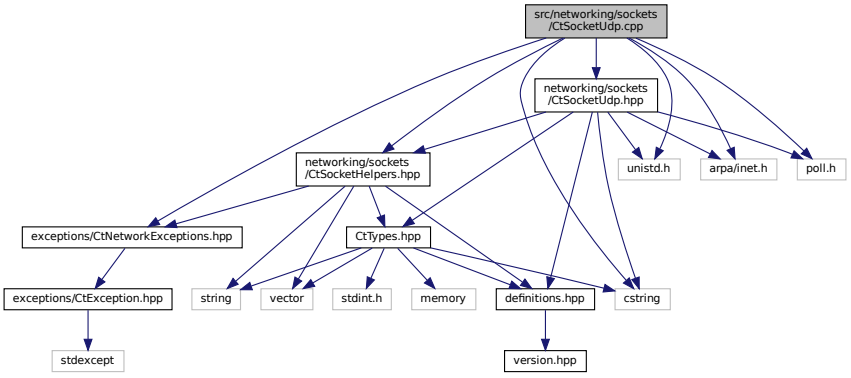
Date

21-01-2024

6.34 src/networking/sockets/CtSocketUdp.cpp File Reference

```
#include "networking/sockets/CtSocketUdp.hpp"
#include "networking/sockets/CtSocketHelpers.hpp"
#include "exceptions/CtNetworkExceptions.hpp"
#include <cstring>
#include <unistd.h>
#include <arpa/inet.h>
#include <poll.h>
```

Include dependency graph for CtSocketUdp.cpp:



6.34.1 Detailed Description

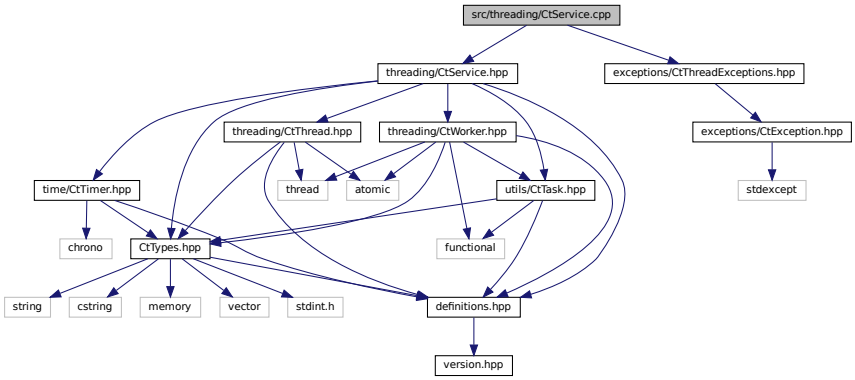
Date

18-01-2024

6.35 src/threading/CtService.cpp File Reference

```
#include "threading/CtService.hpp"
#include "exceptions/CtThreadExceptions.hpp"
```

Include dependency graph for CtService.cpp:



6.35.1 Detailed Description

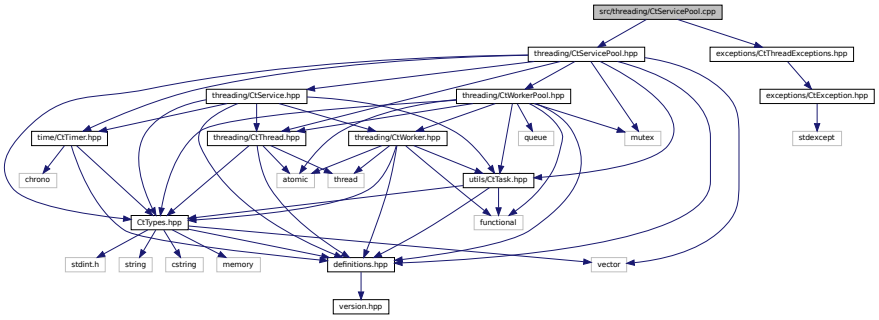
Date

18-01-2024

6.36 src/threading/CtServicePool.cpp File Reference

```
#include "threading/CtServicePool.hpp"
#include "exceptions/CtThreadExceptions.hpp"
```

Include dependency graph for CtServicePool.cpp:



6.36.1 Detailed Description

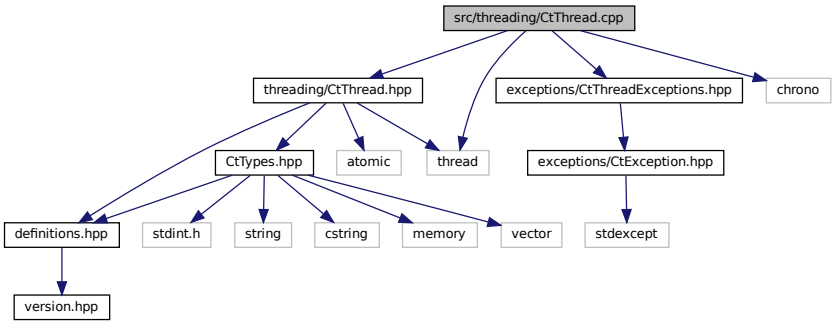
Date

18-01-2024

6.37 src/threading/CtThread.cpp File Reference

```
#include "threading/CtThread.hpp"
#include "exceptions/CtThreadExceptions.hpp"
#include <chrono>
#include <thread>
```

Include dependency graph for CtThread.cpp:



6.37.1 Detailed Description

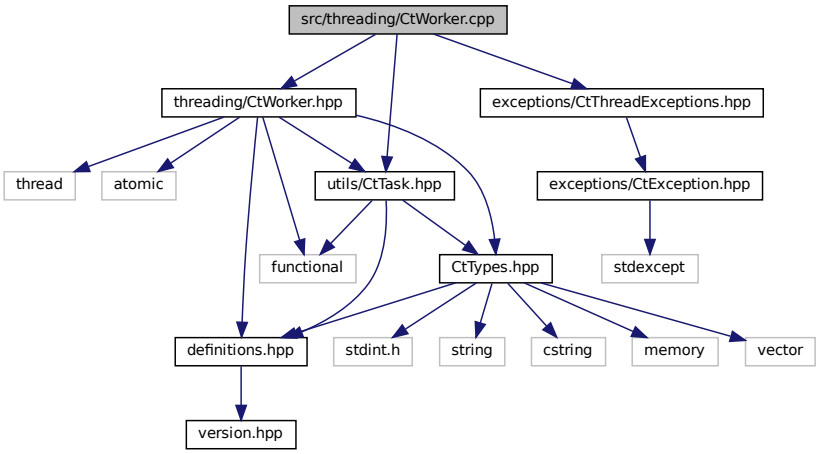
Date

18-01-2024

6.38 src/threading/CtWorker.cpp File Reference

```
#include "threading/CtWorker.hpp"
#include "utils/CtTask.hpp"
```

```
#include "exceptions/CtThreadExceptions.hpp"
```
Include dependency graph for CtWorker.cpp:



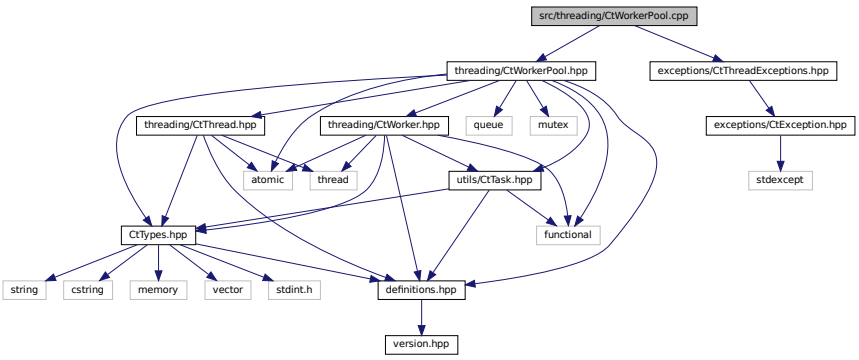
6.38.1 Detailed Description

Date

18-01-2024

6.39 src/threading/CtWorkerPool.cpp File Reference

```
#include "threading/CtWorkerPool.hpp"
#include "exceptions/CtThreadExceptions.hpp"
```
Include dependency graph for CtWorkerPool.cpp:



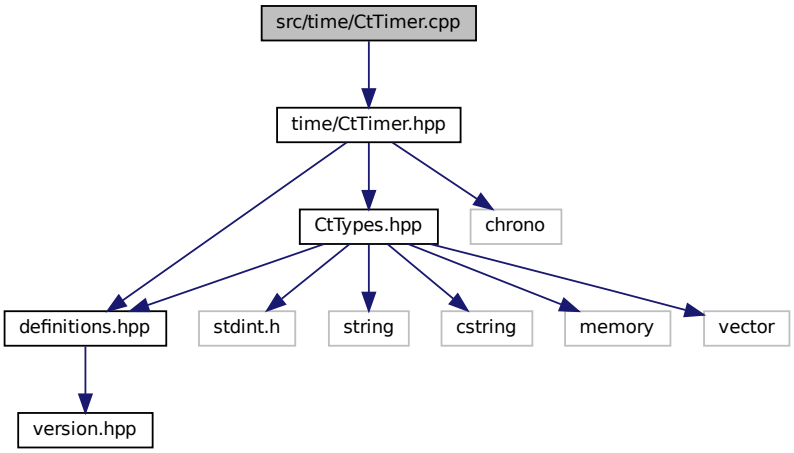
6.39.1 Detailed Description

Date

18-01-2024

6.40 src/time/CtTimer.cpp File Reference

```
#include "time/CtTimer.hpp"
```
Include dependency graph for CtTimer.cpp:



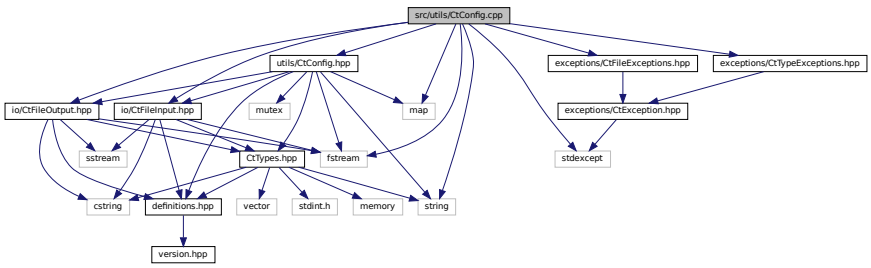
6.40.1 Detailed Description

Date

18-01-2024

6.41 src/utils/CtConfig.cpp File Reference

```
#include "utils/CtConfig.hpp"
#include "io/CtFileInput.hpp"
#include "io/CtFileOutput.hpp"
#include "exceptions/CtFileExceptions.hpp"
#include "exceptions/CtTypeExceptions.hpp"
#include <string>
#include <map>
#include <fstream>
#include <stdexcept>
```
Include dependency graph for CtConfig.cpp:



6.41.1 Detailed Description

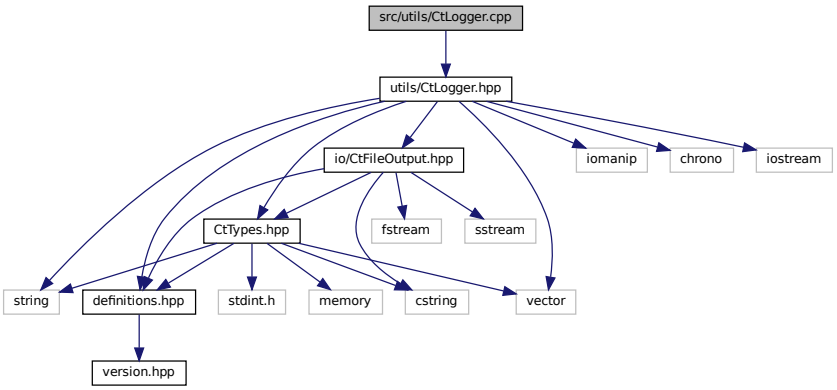
Date

10-03-2024

6.42 src/utils/CtLogger.cpp File Reference

```
#include "utils/CtLogger.hpp"
```

Include dependency graph for CtLogger.cpp:



6.42.1 Detailed Description

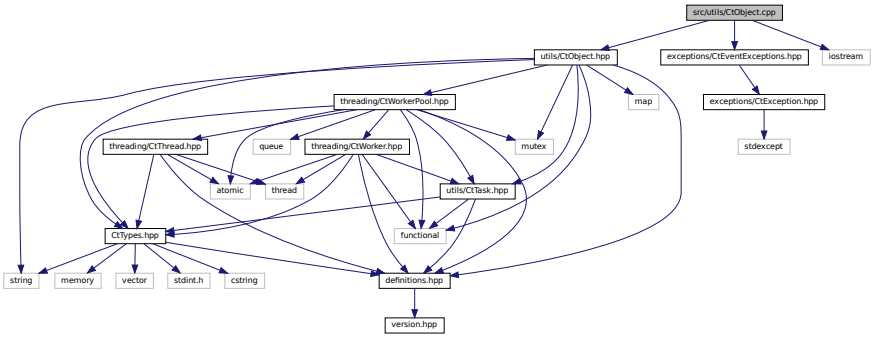
Date

10-03-2024

6.43 src/utils/CtObject.cpp File Reference

```
#include "utils/CtObject.hpp"
#include "exceptions/CtEventExceptions.hpp"
#include <iostream>
```

Include dependency graph for CtObject.cpp:



6.43.1 Detailed Description

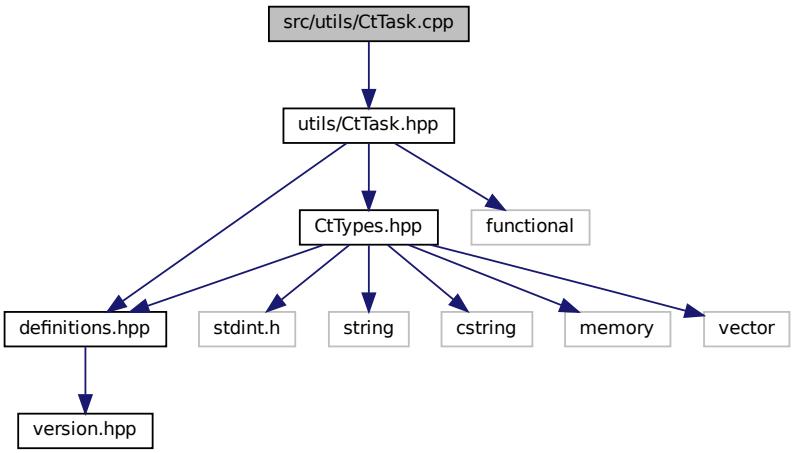
Date

02-02-2024

6.44 src/utils/CtTask.cpp File Reference

```
#include "utils/CtTask.hpp"
```

Include dependency graph for CtTask.cpp:



6.44.1 Detailed Description

Date

18-01-2024

Index