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. Run CMake to configure the project: `cmake -B build -DCMAKE_BUILD_TYPE=Release`
5. Build the project using Make: `cmake --build build --config Release`
6. The compiled binaries will be located in the `build` directory.
7. Run the tests: `ctest --test-dir build -C Release`

Chapter 2

Functional Requirements

2.1 CtCore (001)

2.1.1 CtExceptions (001)

ID	Description
FR-001-001-001	`CtException` must inherit `std::exception` and provide a mechanism to print meaningful messages to the developer.
FR-001-001-002	`CtException` must be inherited by all other exceptions used in CppToolkit.
FR-001-001-003	`CtException` and the inherited classes must provide a method to allocate and initialize its resources.
FR-001-001-004	`CtTypeParseError` should thrown if a type requested cannot be parsed.
FR-001-001-005	`CtKeyNotFoundError` should thrown if a rquested key is not linked to a value.
FR-001-001-006	`CtOutOfRangeError` should thrown if a rquested resource is out of range.
FR-001-001-007	`CtThreadError` should thrown if a `CtThread` started while it is already running.
FR-001-001-008	`CtWorkerError` should thrown if a `CtWorker` or a `CtWorkerPool` started while it is already running.
FR-001-001-009	`CtServiceError` should thrown if a `CtService` started while it is already running.
FR-001-001-010	`CtSocketError` should thrown if a socket allocation failed.
FR-001-001-011	`CtSocketBindError` should thrown if a network interface or a port is not available.
FR-001-001-012	`CtSocketPollError` should thrown if the polling to an already initialized socket failed.
FR-001-001-013	`CtSocketReadError` should thrown if reading of an already initialized socket failed.
FR-001-001-014	`CtSocketWriteError` should thrown if writing of an already initialized socket failed.
FR-001-001-015	`CtFileReadError` should thrown if reading of a file failed.
FR-001-001-016	`CtFileWriteError` should thrown if writing to a file failed.
FR-001-001-017	`CtFileParseError` should thrown if parsing an already open file failed.
FR-001-001-018	`CtEventAlreadyExistsError` should thrown if a `CtObject` try to register an already registered event.
FR-001-001-019	`CtEventNotExistsError` should thrown if an event is not registered to a `CtObject` but connection or triggering called.

2.1.2 CtHelpers (002)

ID	Description
FR-001-002-001	`CtStringHelpers` namespace must provide functions related to string manipulation.
FR-001-002-002	`CtStringHelpers` namespace must provide function `split` that splits a string into substrings given a delimiter.
FR-001-002-003	`CtStringHelpers` namespace must provide function `trim` that removes whitespaces from boths ends of a string.
FR-001-002-004	`CtStringHelpers` namespace must provide functions `StrToInt` for string to int transformation.
FR-001-002-005	`CtStringHelpers` namespace must provide functions `StrToUInt` for string to uint transformation.
FR-001-002-006	`CtStringHelpers` namespace must provide functions `StrToFloat` for string to float transformation.
FR-001-002-007	`CtStringHelpers` namespace must provide functions `StrToDouble` for string to double transformation.
FR-001-002-100	`CtSocketHelpers` namespace must provide a function for setting the timeout for poll requests.
FR-001-002-101	`CtSocketHelpers` namespace must provide a function for getting all available network interfaces.
FR-001-002-102	`CtSocketHelpers` namespace must provide a function for getting the address of a specific network interface.
FR-001-002-103	`CtSocketHelpers` namespace must provide functions for transforming an interface from string to uint and vice versa.

2.1.3 CtTypes (003)

ID	Description
FR-001-003-001	`CppToolkit` core must offer a data type for handling network addresses.
FR-001-003-002	`CppToolkit` core must offer a data type `CtRawData` to store and handle data on any type.
FR-001-003-003	`CtRawData` must provide a method to allocate and initialize its resources.
FR-001-003-004	`CtRawData` should have a default internal buffer size of `2048` bytes if there is no size given by the user.
FR-001-003-005	`CtRawData` must provide the functionality to be allocated by another `CtRawData` object.
FR-001-003-006	`CtRawData` must provide a method to free its resources.
FR-001-003-007	`CtRawData` must maintain the size of the data stored and the max size of the data that can be stored.
FR-001-003-008	`CtRawData` must provide a method to set the following byte.
FR-001-003-009	`CtRawData` must provide a method to set the following number of bytes given a `CtUInt8*` buffer and its size.
FR-001-003-010	`CtRawData` must provide a method to return a pointer to the last N bytes of the internal buffer.
FR-001-003-011	`CtRawData` must provide a method to remove the last N bytes of the internal buffer and modify the size of the data stored accordingly.
FR-001-003-012	`CtRawData` must provide a method to return the size of the buffer.
FR-001-003-013	`CtRawData` must provide a method to return the max size of the buffer.
FR-001-003-014	`CtRawData` must provide a method to return a pointer to the internal buffer.
FR-001-003-015	`CtRawData` must provide a method to copy data to the internal buffer using another `CtRawData` object.
FR-001-003-016	`CtRawData` must provide a method to copy data to the internal buffer using a `CtUInt*` buffer and its size.
FR-001-003-017	`CtRawData` must provide a method to reset the internal buffer to zero size.
FR-001-003-018	`CtRawData` must provide assigment operator.

ID	Description
FR-001-003-019	`CtRawData` must throw `CtOutOfRangeError` if any of its methods try to access a memory out of internal buffer size.

2.2 IO (002)

2.2.1 CtFileInput (001)

ID	Description
FR-002-001-001	`CtFileInput` must open a file given its name.
FR-002-001-002	`CtFileInput` must provide a method to allocate and initialize its resources.
FR-002-001-003	`CtFileInput` must throw `CtFileReadError` if the file cannot be open.
FR-002-001-004	`CtFileInput` must provide a method to free its resources and close the file.
FR-002-001-005	`CtFileInput` must provide a method to read the file.
FR-002-001-006	`CtFileInput` must provide a method to set a delimiter. This delimiter will be used during read to split file into batches.
FR-002-001-007	`CtFileInput` must use the delimiter provided to split the file data to batches during read method call.
FR-002-001-008	If the delimiter size is zero or the delimiter provided is null the read method call must handle the rest of the file as one batch.
FR-002-001-009	The read method of `CtFileInput` must get as argument a `CtRawData` and fill it with the next batch of data or till the buffer is full.
FR-002-001-010	The read method must return `FALSE` in case of end-of-file or `TRUE` in any other case.
FR-002-001-011	`CtFileReadError` must be thrown during read method if the file is not open.

2.2.2 CtFileOutput (002)

ID	Description
FR-002-002-001	`CtFileOutput` must open a file given its name and the writing mode.
FR-002-002-002	`CtFileOutput` must provide a method to allocate and initialize its resources.
FR-002-002-003	Writing mode must be `Append` or `Truncate`. `Append` must append data to a file or create a new one. `Truncate` must create a new file.
FR-002-002-004	`CtFileOutput` must throw `CtFileWriteError` if the file cannot be open.
FR-002-002-005	`CtFileOutput` must provide a method to free its resources and close the file.
FR-002-002-006	`CtFileOutput` must provide a method to set a delimiter.
FR-002-002-007	`CtFileOutput` must provide a method to write data to the file and append the delimiter to them.
FR-002-002-008	If delimiter size is zero or the delimiter provided is null the write method call must write just the data requested with no delimiter.
FR-002-002-009	`CtFileOutput` must provide a method to write data to the file without appending a delimiter to them.
FR-002-002-010	`CtFileWriteError` must be thrown during write method if the file is not open.

2.3 Time (003)

2.3.1 CtTimer (001)

ID	Description
FR-003-001-001	CtTimer must provide a method to allocate and initialize its resources.
FR-003-001-002	CtTimer must provide a method to free its resources.
FR-003-001-003	CtTimer must provide a method for setting the reference point in millisecons.
FR-003-001-004	CtTimer must provide a method for getting the time passed since the reference point in millisecons.
FR-003-001-005	CtTimer must provide a method for getting the milliseconds passed since epoch.

2.4 Utils (004)

2.4.1 CtConfig (001)

ID	Description
FR-004-001-001	CtConfig must provide a method to allocate and initialize its resources given a filename.
FR-004-001-002	CtConfig must provide a method to free its resources.
FR-004-001-003	CtConfig must provide the functionality to save and restore uint, int, float, double and string values to a file.
FR-004-001-004	CtConfig must maintain and provide an internal map key-value set with varibale's names and values that restored or will be stored.
FR-004-001-005	CtConfig must provide a method to write a configuration loaded in the internal map to a file.
FR-004-001-006	CtConfig must provide a method to read a configuration from a file and load it to the internal map.
FR-004-001-007	CtFileReadError must be thrown if an error occur with external file while reading.
FR-004-001-008	CtFileWriteError must be thrown if an error occur with external file while writing.
FR-004-001-009	CtFileParseError must be thrown if an error occur parsing the configuration file.
FR-004-001-010	CtConfig must provide a method for parsing values stored in the map given the key.
FR-004-001-011	CtConfig must provide a method for writing values to the map given the key and the value.
FR-004-001-012	CtKeyNotFoundError must be thrown if a key requested cannot be found in the internal map.
FR-004-001-013	CtTypeParseError must be thrown if a value can not be parsed with the requested type.
FR-004-001-014	CtConfig must provide a method for reseting map deleting all stored values.

2.4.2 CtLogger (002)

ID	Description
FR-004-002-001	CtLogger must provide a method to allocate and initialize its resources given the Log level and a name or an identifier for the logger.
FR-004-002-002	CtLogger must provide a method to free its resources.
FR-004-002-003	CtLogger must provide 5 log levels DEBUG, INFO, WARNING, ERROR, CRITICAL.
FR-004-002-004	CtLogger must provide methods to log a message for all the log levels.
FR-004-002-005	CtLogger must log identifier, log time, log level and the message if the message level is higher or equal to logger level.

2.4.3 CtObject (003)

ID	Description
FR-004-003-001	`CtObject` should be inherited by other classes to utilize event capabilities.
FR-004-003-002	`CtObject` must provide a method to allocate and initialize its resources.
FR-004-003-003	`CtObject` must provide a method to free its resources.
FR-004-003-004	`CtObject` must provide a method to register an event.
FR-004-003-005	`CtEventAlreadyExistsError` must be thrown during register event if the event is already registered.
FR-004-003-006	`CtObject` must provide a method to connect an event with a `CtTask` or a function.
FR-004-003-007	`CtObject` must provide a method to trigger an event.
FR-004-003-008	`CtEventNotExistsError` must be thrown during connect or trigger event if an event does not exists.
FR-004-003-009	`CtObject` must wait for all running activities to stop before free.
FR-004-003-010	`CtObject` must provide a method to wait for all events to run the assigned tasks.

2.5 Threading (005)

2.5.1 CtTask (001)

ID	Description
FR-005-001-001	`CtTask` must maintain and provide a function call and a callback function either by using another `CtTask` or using functions.
FR-005-001-002	`CtTask` must provide a method to allocate and initialize its resources.
FR-005-001-003	`CtTask` must provide a method to free its resources.
FR-005-001-004	`CtTask` must provide a method to get the task function.
FR-005-001-005	`CtTask` must provide a method to get the callback of the task function.
FR-005-001-006	`CtTask` must provide a method to set the task function.
FR-005-001-007	`CtTask` must provide a method to set the callback of the task function.
FR-005-001-008	`CtTask` must provide the assigment operator.

2.5.2 CtThread (002)

ID	Description
FR-005-002-001	`CtThread` must manage and host a std::thread.
FR-005-002-002	`CtThread` should be inherited by other classes to utilize thread capabilities.
FR-005-002-003	`CtThread` must provide a method to allocate and initialize its resources.
FR-005-002-004	`CtThread` must provide and maintain the status of the thread.
FR-005-002-005	`CtThread` must provide a method to get the status of the thread in a thread-safe way.
FR-005-002-006	`CtThread` must provide a method to set the status of the thread in a thread-safe way.
FR-005-002-007	`CtThread` must provide a method to start the thread.
FR-005-002-008	`CtThread` must throw `CtThreadError` during start method if the thread is already running.
FR-005-002-009	`CtThread` must provide a method to stop the thread.
FR-005-002-010	`CtThread` during stop method must change the status of the thread to not running and wait for the thread to finish.
FR-005-002-011	`CtThread` must provide a method that waits for thread exit and to free its resources.
FR-005-002-012	`CtThread` must provide a method to join the thread.
FR-005-002-013	`CtThread` must provide a static method to put the current running thread to sleep for a specified duration in milliseconds.

2.5.3 CtWorker (003)

ID	Description
FR-005-003-001	CtWorker must handle the execution of a CtTask or a function asynchronously.
FR-005-003-002	CtWorker must provide a method to allocate and initialize its resources.
FR-005-003-003	CtWorker must provide a method to free its resources.
FR-005-003-004	CtWorker must provide a method to get the status of the worker in a thread-safe way.
FR-005-003-005	CtWorker must provide a method to set the status of the worker in a thread-safe way.
FR-005-003-006	CtWorker must provide a method to join the task execution.
FR-005-003-007	CtWorker must provide a method to set the task of the worker.
FR-005-003-008	The task of the worker must be set either by CtTask object or by function.
FR-005-003-009	CtWorker must provide the functionality of a callback function call uppon task completion.
FR-005-003-010	CtWorker must provide a method to run the setted task.
FR-005-003-011	CtWorker must throw CtWorkerError during the call of either run or set task if the worker is currenlty running another task.

2.5.4 CtWorkerPool (004)

ID	Description
FR-005-004-001	CtWorkerPool must handle the execution of CtTask objects or functions asynchronously.
FR-005-004-002	CtWorkerPool must maintain a vector of CtWorker objetcs and a queue of CtTask objects.
FR-005-004-003	CtWorkerPool must provide a method to allocate and initialize its resources given the number of CtWorker that need to be utilized.
FR-005-004-004	CtWorkerPool must provide a method to free its resources.
FR-005-004-005	CtWorkerPool must wait for all running activities to stop before closing.
FR-005-004-006	CtWorkerPool must provide a method to add a new task to the queue either by CtTask or by function in a thread-safe way.
FR-005-004-007	CtWorkerPool must provide a method to join the execution of all active and queued tasks.
FR-005-004-008	When a CtWorker is free, the next available active task must be assigned to it in a thread-safe way.
FR-005-004-009	If no active or queued tasks are available the CtWorkerPool should stop running till a new task is added.

2.5.5 CtService (005)

ID	Description
FR-005-005-001	CtService must handle the execution of a CtTask object or a function asynchronously and repeatedly.
FR-005-005-002	CtService must provide a method to allocate and initialize all resources needed to support its functionality.
FR-005-005-003	CtService must provide a method to free its resources.
FR-005-005-004	CtService must wait for all running activities to stop before closing.
FR-005-005-005	CtService must provide a method to start running the service.
FR-005-005-006	CtService must provide a method to stop running the service.
FR-005-005-007	CtService must throw CtServiceError if a service is started more than once.
FR-005-005-008	CtService should inherit the CtThread and run the given tasks repeatedly at constant rates.

ID	Description
FR-005-005-009	`CtService` must maintain a counter representing the number of task executions skipped because they were not yet completed.
FR-005-005-010	`CtService` must maintain a counter representing the number of tasks that should have been executed.
FR-005-005-011	`CtService` must provide a method to get the ratio of skipped task executions to the total task executions.
FR-005-005-012	`CtService` must maintain a variable that represents the minimum time interval between task executions in milliseconds.
FR-005-005-013	The interval given for each service must be calculated as an integer multiplier of this minimum interval.

2.5.6 CtServicePool (006)

ID	Description
FR-005-006-001	`CtServicePool` must handle the execution of `CtTask` objects or functions asynchronously and repeatedly.
FR-005-006-002	`CtServicePool` must provide a method to allocate and initialize all resources needed to support its functionality.
FR-005-006-003	`CtServicePool` must provide a method to free its resources.
FR-005-006-004	`CtServicePool` must wait for all running activities to stop before closing.
FR-005-006-005	`CtServicePool` must provide a method to add a service task either by `CtTask` or by function a call in a thread-safe way.
FR-005-006-006	`CtServicePool` must provide a method to remove a service task either by `CtTask` or by function a call in a thread-safe way.
FR-005-006-007	`CtServicePack` a struct must be used to store tasks' period in number of intervals, identifier and task itself.
FR-005-006-008	`CtServicePool` must maintain an internal vector of `CtServicePack` objects.
FR-005-006-009	`CtServicePool` must provide a method for start running the services.
FR-005-006-010	`CtServicePool` must provide a method for stop running the services.
FR-005-006-011	`CtServicePool` should inherit the `CtThread` and run the given tasks repeatedly at constant rates.

2.6 Networking (006)

2.6.1 CtSocketUdp (001)

ID	Description
FR-006-001-001	`CtSocketUdp` must provide a method to allocate and initialize all resources needed to support its functionality.
FR-006-001-002	`CtSocketUdp` must provide a method to free its resources.
FR-006-001-003	`CtSocketError` must be thrown if allocation of `CtSocketUdp` failed.
FR-006-001-004	`CtSocketUdp` must provide a method for configuring a socket to subscribe for data.
FR-006-001-005	`CtSocketUdp` must provide a method for configuring a socket to publish data.
FR-006-001-006	`CtSocketUdp` must provide a method for polling a socket to read data if available.
FR-006-001-007	`CtSocketUdp` must provide a method for polling a socket availability to publish data.
FR-006-001-008	`CtSocketPollError` must be thrown if polling a socket failed.
FR-006-001-009	`CtSocketUdp` must provide methods to send data over a publisher socket either using `CtRawData` or `CtUInt8*` buffer.

ID	Description
FR-006-001-010	`CtSocketUdp` must provide methods to read data from a subscriber socket either using `CtRawData` or `CtUInt8*` buffer.
FR-006-001-011	`CtSocketWriteError` must be thrown if writing data to a socket failed.
FR-006-001-012	`CtSocketReadError` must be thrown if reading data from a socket failed.

Chapter 3

Namespace Index

3.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 4

Hierarchical Index

4.1 Class Hierarchy

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

Chapter 5

Class Index

5.1 Class List

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

Chapter 6

File Index

6.1 File List

Here is a list of all files with brief descriptions:

Chapter 7

Namespace Documentation

7.1 CtSocketHelpers Namespace Reference

This namespace contains socket helper functions.

Functions

- EXPORTED_API void setSocketTimeout (CtInt32 socketTimeout)

  Set the Socket Timeout object.
- EXPORTED_API CtVector< CtString > getInterfaces ()

  Get all available interfaces the device.
- EXPORTED_API CtString interfaceToAddress (const CtString &p_ifName)

  Get address of a specific interface.
- EXPORTED_API CtUInt32 getAddressAsUInt (const CtString &p_addr)

  Convert address to uin32_t.
- EXPORTED_API CtString getAddressAsString (CtUInt32 p_addr)

  Convert address to CtString.

Variables

- static CtInt32 socketTimeout = 0

7.1.1 Detailed Description

This namespace contains socket helper functions.

7.1.2 Function Documentation

7.1.2.1 getAddressAsString()

```
CtString CtSocketHelpers::getAddressAsString (
            CtUInt32 p_addr )
```

Convert address to CtString.

FR-001-002-103

Parameters

p_addr	The address in form of CtUInt32

Returns

CtString The address in form of CtString

Definition at line 162 of file CtHelpers.cpp.

7.1.2.2 getAddressAsUInt()

```
CtUInt32 CtSocketHelpers::getAddressAsUInt (
            const CtString & p_addr )
```

Convert address to uin32_t.

FR-001-002-103

Parameters

p_addr	The address in form of CtString

Returns

CtUInt32 The address in form of CtUInt32

Definition at line 152 of file CtHelpers.cpp.

7.1.2.3 getInterfaces()

```
CtVector< CtString > CtSocketHelpers::getInterfaces ( )
```

Get all available interfaces the device.

FR-001-002-101

Returns

CtVector<CtString> The available interfaces.

Definition at line 105 of file CtHelpers.cpp.

7.1.2.4 interfaceToAddress()

```
CtString CtSocketHelpers::interfaceToAddress (
            const CtString & p_ifName )
```

Get address of a specific interface.

FR-001-002-102

Parameters

p_ifName	The name of the interface.

Returns

CtString The address of the interface.

Definition at line 119 of file CtHelpers.cpp.

7.1.2.5 setSocketTimeout()

```
void CtSocketHelpers::setSocketTimeout (
            CtInt32 socketTimeout )
```

Set the Socket Timeout object.

FR-001-002-100

Parameters

socketTimeout	The target timeout for the poll request.

Definition at line 101 of file CtHelpers.cpp.

7.1.3 Variable Documentation

7.1.3.1 socketTimeout

```
CtInt32 CtSocketHelpers::socketTimeout = 0  [static]
```

The timeout value for socket poll operations.

Definition at line 125 of file CtHelpers.hpp.

7.2 CtStringHelpers Namespace Reference

This namespace contains string helper functions.

Functions

- EXPORTED_API void split (const CtString &p_string, char p_delimiter, CtVector< CtString > *p_result)

  This method splits the string into substrings using the given delimiter.
- EXPORTED_API CtString trim (const CtString &p_string)

  This method trims the string from the left and right side.
- EXPORTED_API CtDouble StrToDouble (const CtString &p_str)

  This method converts a string to a CtDouble.
- EXPORTED_API CtFloat StrToFloat (const CtString &p_str)

  This method converts a string to a CtFloat.
- EXPORTED_API CtUInt32 StrToUInt (const CtString &p_str)

  This method converts a string to a CtUInt32.
- EXPORTED_API CtInt32 StrToInt (const CtString &p_str)

  This method converts a string to a CtInt32.

7.2.1 Detailed Description

This namespace contains string helper functions.

FR-001-002-001

7.2.2 Function Documentation

7.2.2.1 split()

```
EXPORTED_API void CtStringHelpers::split (
            const CtString & p_string,
            char p_delimiter,
            CtVector< CtString > * p_result )
```

This method splits the string into substrings using the given delimiter.

FR-001-002-002

Parameters

p_string	The string to be split.
p_delimiter	This is the delimiter that will be used to split the string.
p_result	The vector that will contain the substrings.

7.2.2.2 StrToDouble()

```
CtDouble CtStringHelpers::StrToDouble (
            const CtString & p_str )
```

This method converts a string to a CtDouble.

FR-001-002-007

If the string can not be converted to a CtDouble, a CtTypeParseError exception is thrown.

Parameters

p_str	The string to be converted.

Returns

CtDouble The converted CtDouble.

Definition at line 61 of file CtHelpers.cpp.

7.2.2.3 StrToFloat()

```
CtFloat CtStringHelpers::StrToFloat (
            const CtString & p_str )
```

This method converts a string to a CtFloat.

FR-001-002-006

If the string can not be converted to a CtFloat, a CtTypeParseError exception is thrown.

Parameters

p_str	The string to be converted.

Returns

CtFloat The converted CtFloat.

Definition at line 71 of file CtHelpers.cpp.

7.2.2.4 StrToInt()

```
CtInt32 CtStringHelpers::StrToInt (
            const CtString & p_str )
```

This method converts a string to a CtInt32.

FR-001-002-004

If the string can not be converted to a CtInt32, a CtTypeParseError exception is thrown.

Parameters

p_str	The string to be converted.

Returns

CtInt32 The converted CtInt32.

Definition at line 91 of file CtHelpers.cpp.

7.2.2.5 StrToUInt()

```
CtUInt32 CtStringHelpers::StrToUInt (
            const CtString & p_str )
```

This method converts a string to a CtUInt32.

FR-001-002-005

If the string can not be converted to a CtUInt32, a CtTypeParseError exception is thrown.

Parameters

p_str	The string to be converted.

Returns

CtUInt32 The converted CtUInt32.

Definition at line 81 of file CtHelpers.cpp.

7.2.2.6 trim()

```
CtString CtStringHelpers::trim (
            const CtString & p_string )
```

This method trims the string from the left and right side.

FR-001-002-003

Parameters

p_string	The string to be trimmed.

Returns

CtString The trimmed string.

Definition at line 55 of file CtHelpers.cpp.

Chapter 8

Class Documentation

8.1 _CtNetAddress Struct Reference

Struct describing a network address.

`#include <CtTypes.hpp>`

Public Attributes

- CtString addr
- CtUInt16 port

8.1.1 Detailed Description

Struct describing a network address.

FR-001-003-001

The network address is described by the IP address and the port number.

Definition at line 91 of file CtTypes.hpp.

8.1.2 Member Data Documentation

8.1.2.1 addr

`CtString _CtNetAddress::addr`

Definition at line 92 of file CtTypes.hpp.

8.1.2.2 port

```
CtUInt16 _CtNetAddress::port
```

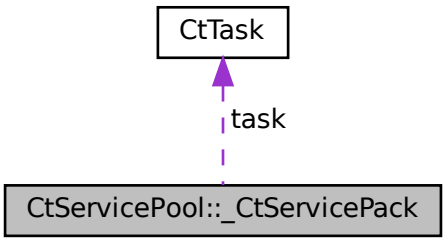
Definition at line 93 of file CtTypes.hpp.

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

- include/core/CtTypes.hpp

8.2 CtServicePool::_CtServicePack Struct Reference

Collaboration diagram for CtServicePool::_CtServicePack:



Public Attributes

- CtTask task
- CtString id
- CtUInt32 nslots

8.2.1 Detailed Description

Definition at line 78 of file CtServicePool.hpp.

8.2.2 Member Data Documentation

8.2.2.1 id

```
CtString CtServicePool::_CtServicePack::id
```

Definition at line 80 of file CtServicePool.hpp.

8.2.2.2 nslots

```
CtUInt32 CtServicePool::_CtServicePack::nslots
```

Definition at line 81 of file CtServicePool.hpp.

8.2.2.3 task

```
CtTask CtServicePool::_CtServicePack::task
```

Definition at line 79 of file CtServicePool.hpp.

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

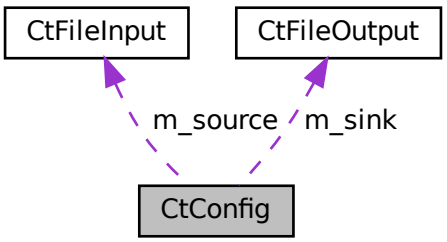
- include/threading/CtServicePool.hpp

8.3 CtConfig Class Reference

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

```
#include <CtConfig.hpp>
```

Collaboration diagram for CtConfig:



Public Member Functions

- EXPORTED_API CtConfig (const CtString &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 CtInt32 parseAsInt (const CtString &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 CtUInt32 parseAsUInt (const CtString &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 CtFloat parseAsFloat (const CtString &p_key)

  Parse a value as a CtFloat or throw CtKeyNotFoundError if key is not found in the map or throw CtParseError if key value cannot be parsed as CtFloat.
- EXPORTED_API CtDouble parseAsDouble (const CtString &p_key)

  Parse a value as a CtDouble-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 CtDouble.
- EXPORTED_API CtString parseAsString (const CtString &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 CtString &p_key, const CtInt32 &p_value)

  Write value to key as int.
- EXPORTED_API void writeUInt (const CtString &p_key, const CtUInt32 &p_value)

  Write value to key as uint.
- EXPORTED_API void writeFloat (const CtString &p_key, const CtFloat &p_value)

  Write value to key as CtFloat.
- EXPORTED_API void writeDouble (const CtString &p_key, const CtDouble &p_value)

  Write value to key as CtDouble.
- EXPORTED_API void writeString (const CtString &p_key, const CtString &p_value)

  Write value to key as string.
- EXPORTED_API void reset ()

  This method resets the configuration values.

Private Member Functions

- CtString getValue (const CtString &p_key)

  This method returns the value assosiated with the given key or throw CtKeyNotFoundError if key is not found in the map.
- void parseLine (const CtString &p_line)

  This method gets a line as input and parse it in order to find the key and value of configured item. These values are stored in the CtMap m_configValues. This method can throw CtFileParseError if file cannot be parsed.

Private Attributes

- CtMutex m_mtx_control
- CtFileInput * m_source
- CtFileOutput * m_sink
- CtString m_configFile
- CtMap< CtString, CtString > m_configValues

8.3.1 Detailed Description

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

The CtConfig class provides a mechanism for reading and writing configuration files providing key-value pairs of various data types. The class can parse integer, unsigned integer, CtFloat, CtDouble, and string values. The class is thread-safe and can be used in multi-threaded environments.

Definition at line 49 of file CtConfig.hpp.

8.3.2 Constructor & Destructor Documentation

8.3.2.1 CtConfig()

```
CtConfig::CtConfig (
            const CtString & p_configFile )  [explicit]
```

Constructor for CtConfig.

FR-004-001-001

Parameters

configFile	The path to the configuration file to be parsed.

Definition at line 34 of file CtConfig.cpp.

8.3.2.2 ~CtConfig()

```
CtConfig::~CtConfig ( )
```

Destructor for cleaning up resources.

FR-004-001-002

Definition at line 39 of file CtConfig.cpp.

8.3.3 Member Function Documentation

8.3.3.1 getValue()

```
CtString CtConfig::getValue (
            const CtString & p_key )  [private]
```

This method returns the value assosiated with the given key or throw CtKeyNotFoundError if key is not found in the map.

FR-004-001-004 FR-004-001-012

Parameters

key	The key value to be parsed.

Returns

CtString The string value assosiated with the given key.

Definition at line 140 of file CtConfig.cpp.

8.3.3.2 parseAsDouble()

```
CtDouble CtConfig::parseAsDouble (
            const CtString & p_key )
```

Parse a value as a CtDouble-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 CtDouble.

FR-004-001-010 FR-004-001-003 FR-004-001-013

Parameters

key	The key value to be parsed.

Returns

The parsed CtDouble value.

Definition at line 127 of file CtConfig.cpp.

8.3.3.3 parseAsFloat()

```
CtFloat CtConfig::parseAsFloat (
            const CtString & p_key )
```

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

FR-004-001-010 FR-004-001-003 FR-004-001-013

Parameters

key	The key value to be parsed.

Returns

The parsed floating-point value.

Definition at line 122 of file CtConfig.cpp.

8.3.3.4 parseAsInt()

```
CtInt32 CtConfig::parseAsInt (
            const CtString & 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.

FR-004-001-010 FR-004-001-003 FR-004-001-013

Parameters

key	The key value to be parsed.

Returns

The parsed integer value.

Definition at line 112 of file CtConfig.cpp.

8.3.3.5 parseAsString()

```
CtString CtConfig::parseAsString (
            const CtString & p_key )
```

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

FR-004-001-010 FR-004-001-003

Parameters

key	The key value to be parsed.

Returns

The parsed string.

Definition at line 132 of file CtConfig.cpp.

8.3.3.6 parseAsUInt()

```
CtUInt32 CtConfig::parseAsUInt (
            const CtString & 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.

FR-004-001-010 FR-004-001-003 FR-004-001-013

Parameters

key	The key value to be parsed.

Returns

The parsed unsigned integer value.

Definition at line 117 of file CtConfig.cpp.

8.3.3.7 parseLine()

```
void CtConfig::parseLine (
            const CtString & p_line )  [private]
```

This method gets a line as input and parse it in order to find the key and value of configured item. These values are stored in the CtMap m_configValues. This method can throw CtFileParseError if file cannot be parsed.

FR-004-001-006 FR-004-001-009

Parameters

line	

Definition at line 82 of file CtConfig.cpp.

8.3.3.8 read()

```
void CtConfig::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.

FR-004-001-006 FR-004-001-009 FR-004-001-007

Definition at line 48 of file CtConfig.cpp.

8.3.3.9 reset()

```
void CtConfig::reset ( )
```

This method resets the configuration values.

FR-004-001-014

Returns

void

Definition at line 136 of file CtConfig.cpp.

8.3.3.10 write()

```
void CtConfig::write ( )
```

Write data to config file.

FR-004-001-005 FR-004-001-008

Definition at line 64 of file CtConfig.cpp.

8.3.3.11 writeDouble()

```
void CtConfig::writeDouble (
            const CtString & p_key,
            const CtDouble & p_value )
```

Write value to key as CtDouble.

FR-004-001-011 FR-004-001-003

Parameters

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

Definition at line 160 of file CtConfig.cpp.

8.3.3.12 writeFloat()

```
void CtConfig::writeFloat (
```

```
            const CtString & p_key,
            const CtFloat & p_value )
```

Write value to key as CtFloat.

FR-004-001-011 FR-004-001-003

Parameters

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

Definition at line 156 of file CtConfig.cpp.

8.3.3.13 writeInt()

```
void CtConfig::writeInt (
            const CtString & p_key,
            const CtInt32 & p_value )
```

Write value to key as int.

FR-004-001-011 FR-004-001-003

Parameters

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

Definition at line 148 of file CtConfig.cpp.

8.3.3.14 writeString()

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

Write value to key as string.

FR-004-001-011 FR-004-001-003

Parameters

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

Definition at line 164 of file CtConfig.cpp.

8.3.3.15 writeUInt()

```
void CtConfig::writeUInt (
            const CtString & p_key,
            const CtUInt32 & p_value )
```

Write value to key as uint.

FR-004-001-011 FR-004-001-003

Parameters

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

Definition at line 152 of file CtConfig.cpp.

8.3.4 Member Data Documentation

8.3.4.1 m_configFile

```
CtString CtConfig::m_configFile  [private]
```

The path to the configuration file.

Definition at line 247 of file CtConfig.hpp.

8.3.4.2 m_configValues

```
CtMap<CtString, CtString> CtConfig::m_configValues  [private]
```

A map to store configuration key-value pairs.

Definition at line 248 of file CtConfig.hpp.

8.3.4.3 m_mtx_control

```
CtMutex CtConfig::m_mtx_control  [private]
```

Internal mutex for synchronization.

Definition at line 244 of file CtConfig.hpp.

8.3.4.4 m_sink

```
CtFileOutput* CtConfig::m_sink  [private]
```

The sink file for writing configuration values.

Definition at line 246 of file CtConfig.hpp.

8.3.4.5 m_source

```
CtFileInput* CtConfig::m_source  [private]
```

The source file for reading configuration values.

Definition at line 245 of file CtConfig.hpp.

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

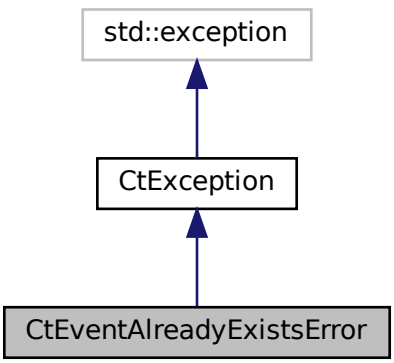
- include/utils/CtConfig.hpp
- src/utils/CtConfig.cpp

8.4 CtEventAlreadyExistsError Class Reference

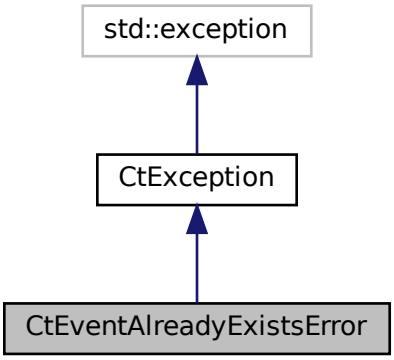
This exception is thrown when an event already exists in the event manager.

`#include <CtEventExceptions.hpp>`

Inheritance diagram for CtEventAlreadyExistsError:



Collaboration diagram for CtEventAlreadyExistsError:



Public Member Functions

- CtEventAlreadyExistsError (const CtString &msg)

Additional Inherited Members

8.4.1 Detailed Description

This exception is thrown when an event already exists in the event manager.

FR-001-001-018 FR-001-001-002 FR-001-001-003

Definition at line 56 of file CtEventExceptions.hpp.

8.4.2 Constructor & Destructor Documentation

8.4.2.1 CtEventAlreadyExistsError()

```
CtEventAlreadyExistsError::CtEventAlreadyExistsError (
            const CtString & msg )  [inline], [explicit]
```

Definition at line 58 of file CtEventExceptions.hpp.

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

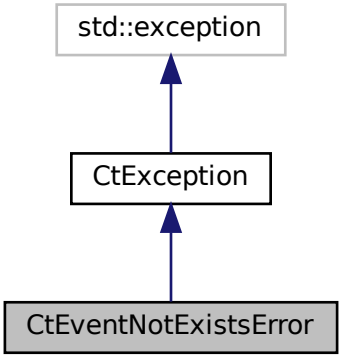
- include/core/exceptions/CtEventExceptions.hpp

8.5 CtEventNotExistsError Class Reference

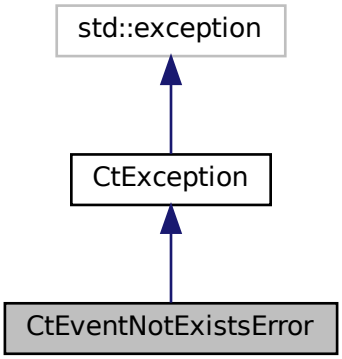
This exception is thrown when an event does not exist in the event manager.

`#include <CtEventExceptions.hpp>`

Inheritance diagram for CtEventNotExistsError:



Collaboration diagram for CtEventNotExistsError:



Public Member Functions

- CtEventNotExistsError (const CtString &msg)

Additional Inherited Members

8.5.1 Detailed Description

This exception is thrown when an event does not exist in the event manager.

FR-001-001-019 FR-001-001-002 FR-001-001-003

Definition at line 44 of file CtEventExceptions.hpp.

8.5.2 Constructor & Destructor Documentation

8.5.2.1 CtEventNotExistsError()

```
CtEventNotExistsError::CtEventNotExistsError (
            const CtString & msg )  [inline], [explicit]
```

Definition at line 46 of file CtEventExceptions.hpp.

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

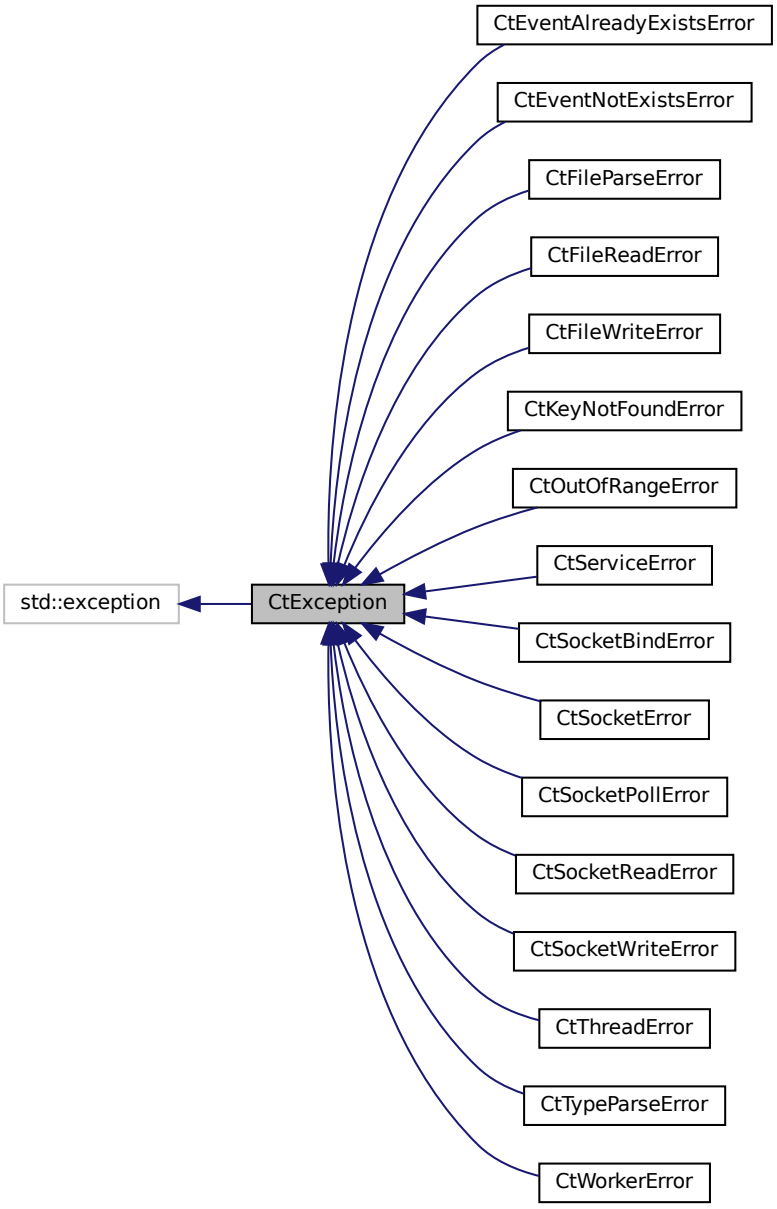
- include/core/exceptions/CtEventExceptions.hpp

8.6 CtException Class Reference

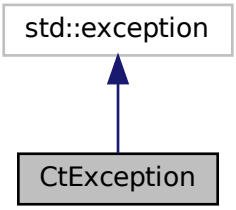
An exception class for the cpptoolkit library.

```
#include <CtException.hpp>
```

Inheritance diagram for CtException:



Collaboration diagram for CtException:



Public Member Functions

- const char * what () const noexcept override
  This method returns the message stored in the exception.

Protected Member Functions

- CtException (const CtString &msg)
  Construct a new Ct Exception object.

Private Attributes

- CtString m_msg

8.6.1 Detailed Description

An exception class for the cpptoolkit library.

This is an abstract class derived from std::exception and is used as a base class for all the exceptions in the library.

Definition at line 47 of file CtException.hpp.

8.6.2 Constructor & Destructor Documentation

8.6.2.1 CtException()

```
CtException::CtException (
            const CtString & msg )  [inline], [explicit], [protected]
```

Construct a new Ct Exception object.

FR-001-001-003

Parameters

msg	Message to be stored in the exception.

Definition at line 56 of file CtException.hpp.

8.6.3 Member Function Documentation

8.6.3.1 what()

```
const char* CtException::what ( ) const  [inline], [override], [noexcept]
```

This method returns the message stored in the exception.

FR-001-001-001

Returns

const char* the message stored in the exception.

Definition at line 66 of file CtException.hpp.

8.6.4 Member Data Documentation

8.6.4.1 m_msg

```
CtString CtException::m_msg  [private]
```

The message stored in the exception.

Definition at line 71 of file CtException.hpp.

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

- include/core/exceptions/CtException.hpp

8.7 CtFileInput Class Reference

CtFileInput class for reading data from file.

```
#include <CtFileInput.hpp>
```

Public Member Functions

- EXPORTED_API CtFileInput (const CtString &p_fileName)

  Constructs the CtFileInput object.
- EXPORTED_API ~CtFileInput ()

  Destructor for CtFileInput.
- EXPORTED_API void setDelimiter (const CtChar *p_delim, CtUInt8 p_delim_size)

  Set the the delimiter of read() method.
- EXPORTED_API CtBool read (CtRawData *p_data)

  This method read data from the file.

Private Attributes

- std::ifstream m_file
- CtChar * m_delim
- CtUInt8 m_delim_size

8.7.1 Detailed Description

CtFileInput class for reading data from file.

This class provides an interface for reading data from a file. The data can be read in batches or one by one.

```
// create a file input object
CtFileInput fileInput("input.txt");
CtRawData data;
while (fileInput.read(&data)) {
   // process data
}
```

Definition at line 57 of file CtFileInput.hpp.

8.7.2 Constructor & Destructor Documentation

8.7.2.1 CtFileInput()

```
CtFileInput::CtFileInput (
            const CtString & p_fileName )  [explicit]
```

Constructs the CtFileInput object.

FR-002-001-001 FR-002-001-002 FR-002-001-003

Parameters

p_fileName	Filename.

Definition at line 34 of file CtFileInput.cpp.

8.7.2.2 ~CtFileInput()

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

Destructor for CtFileInput.

FR-002-001-004

Performs any necessary cleanup.

Definition at line 43 of file CtFileInput.cpp.

8.7.3 Member Function Documentation

8.7.3.1 read()

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

This method read data from the file.

FR-002-001-005 FR-002-001-007 FR-002-001-008 FR-002-001-009 FR-002-001-010 FR-002-001-011

Parameters

p_data	Where to store the data read

Returns

CtBool Returns True on success or False on EOF.

Definition at line 65 of file CtFileInput.cpp.

8.7.3.2 setDelimiter()

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

Set the the delimiter of read() method.

FR-002-001-006

Parameters

p_delim	The delimiter.
p_delim_size	The delimiter size.

Definition at line 52 of file CtFileInput.cpp.

8.7.4 Member Data Documentation

8.7.4.1 m_delim

```
CtChar* CtFileInput::m_delim  [private]
```

Batch read delimiter.

Definition at line 106 of file CtFileInput.hpp.

8.7.4.2 m_delim_size

```
CtUInt8 CtFileInput::m_delim_size  [private]
```

Delimeter size.

Definition at line 107 of file CtFileInput.hpp.

8.7.4.3 m_file

```
std::ifstream CtFileInput::m_file  [private]
```

File stream.

Definition at line 105 of file CtFileInput.hpp.

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

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

8.8 CtFileOutput Class Reference

CtFileOutput class for writing data to file.

```
#include <CtFileOutput.hpp>
```

Public Types

- enum class WriteMode { Append , Truncate }

  Enum representing write mode.

Public Member Functions

- EXPORTED_API CtFileOutput (const CtString &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.
- EXPORTED_API void writePart (CtRawData *p_data)

  This method writes to file.

Private Attributes

- std::ofstream m_file
- std::unique_ptr< char[]> m_delim
- CtUInt8 m_delim_size

8.8.1 Detailed Description

CtFileOutput class for writing data to file.

This class provides an interface for writing data to a file. The data can be written in batches or one by one.

```
// create a file output object
CtFileOutput fileOutput("output.txt");
fileOutput.write("Hello, World!");
```

Definition at line 55 of file CtFileOutput.hpp.

8.8.2 Member Enumeration Documentation

8.8.2.1 WriteMode

```
enum CtFileOutput::WriteMode  [strong]
```

Enum representing write mode.

FR-002-002-003

Enumerator

Append	
Truncate	

Definition at line 62 of file CtFileOutput.hpp.

8.8.3 Constructor & Destructor Documentation

8.8.3.1 CtFileOutput()

```
CtFileOutput::CtFileOutput (
            const CtString & p_fileName,
            WriteMode p_mode = WriteMode::Append )  [explicit]
```

Constructs the CtFileOutput object.

FR-002-002-001 FR-002-002-002 FR-002-002-003 FR-002-002-004

Parameters

p_fileName	Filename.

Definition at line 34 of file CtFileOutput.cpp.

8.8.3.2 ~CtFileOutput()

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

Destructor for CtFileOutput.

FR-002-002-005

Performs any necessary cleanup.

Definition at line 50 of file CtFileOutput.cpp.

8.8.4 Member Function Documentation

8.8.4.1 setDelimiter()

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

Set the the delimiter of write() method.

FR-002-002-006

Parameters

p_delim	The delimiter.
p_delim_size	The delimiter size.

Definition at line 56 of file CtFileOutput.cpp.

8.8.4.2 write()

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

This method writes to file.

FR-002-002-007 FR-002-002-008 FR-002-002-010

Use this method to write data one by one. After writing the data, the delimiter is written.

Parameters

p_data	The data to be written.

Returns

void

Definition at line 68 of file CtFileOutput.cpp.

8.8.4.3 writePart()

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

This method writes to file.

FR-002-002-009 FR-002-002-010

Use this method to write data in batches. No delimiter is written.

Parameters

p_data	The data to be written.

Returns

void

Definition at line 79 of file CtFileOutput.cpp.

8.8.5 Member Data Documentation

8.8.5.1 m_delim

```
std::unique_ptr<char[]> CtFileOutput::m_delim  [private]
```

Batch write delimiter.

Definition at line 126 of file CtFileOutput.hpp.

8.8.5.2 m_delim_size

```
CtUInt8 CtFileOutput::m_delim_size  [private]
```

Delimeter size.

Definition at line 127 of file CtFileOutput.hpp.

8.8.5.3 m_file

```
std::ofstream CtFileOutput::m_file  [private]
```

File stream.

Definition at line 125 of file CtFileOutput.hpp.

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

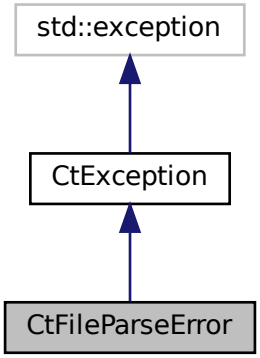
- include/io/CtFileOutput.hpp
- src/io/CtFileOutput.cpp

8.9 CtFileParseError Class Reference

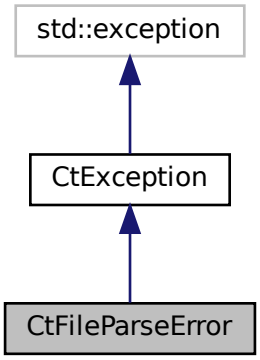
This exception is thrown when a file cannot be parsed.

`#include <CtFileExceptions.hpp>`

Inheritance diagram for CtFileParseError:



Collaboration diagram for CtFileParseError:



Public Member Functions

- CtFileParseError (const CtString &msg)

Additional Inherited Members

8.9.1 Detailed Description

This exception is thrown when a file cannot be parsed.

FR-001-001-017 FR-001-001-002 FR-001-001-003

Definition at line 68 of file CtFileExceptions.hpp.

8.9.2 Constructor & Destructor Documentation

8.9.2.1 CtFileParseError()

```
CtFileParseError::CtFileParseError (
            const CtString & msg )  [inline], [explicit]
```

Definition at line 70 of file CtFileExceptions.hpp.

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

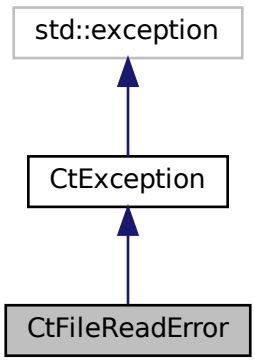
- include/core/exceptions/CtFileExceptions.hpp

8.10 CtFileReadError Class Reference

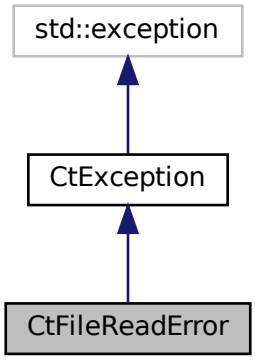
This exception is thrown when a file cannot be read.

```
#include <CtFileExceptions.hpp>
```

Inheritance diagram for CtFileReadError:



Collaboration diagram for CtFileReadError:



Public Member Functions

- CtFileReadError (const CtString &msg)

Additional Inherited Members

8.10.1 Detailed Description

This exception is thrown when a file cannot be read.

FR-001-001-015 FR-001-001-002 FR-001-001-003

Definition at line 44 of file CtFileExceptions.hpp.

8.10.2 Constructor & Destructor Documentation

8.10.2.1 CtFileReadError()

```
CtFileReadError::CtFileReadError (
            const CtString & msg )  [inline], [explicit]
```

Definition at line 46 of file CtFileExceptions.hpp.

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

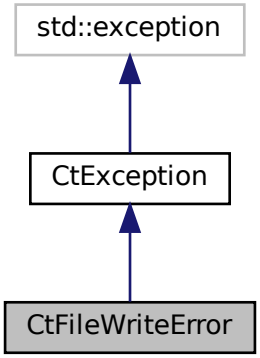
- include/core/exceptions/CtFileExceptions.hpp

8.11 CtFileWriteError Class Reference

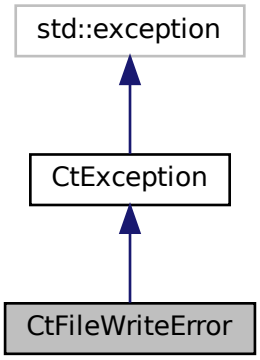
This exception is thrown when a file cannot be written.

`#include <CtFileExceptions.hpp>`

Inheritance diagram for CtFileWriteError:



Collaboration diagram for CtFileWriteError:



Public Member Functions

- CtFileWriteError (const CtString &msg)

Additional Inherited Members

8.11.1 Detailed Description

This exception is thrown when a file cannot be written.

FR-001-001-016 FR-001-001-002 FR-001-001-003

Definition at line 56 of file CtFileExceptions.hpp.

8.11.2 Constructor & Destructor Documentation

8.11.2.1 CtFileWriteError()

```
CtFileWriteError::CtFileWriteError (
            const CtString & msg )  [inline], [explicit]
```

Definition at line 58 of file CtFileExceptions.hpp.

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

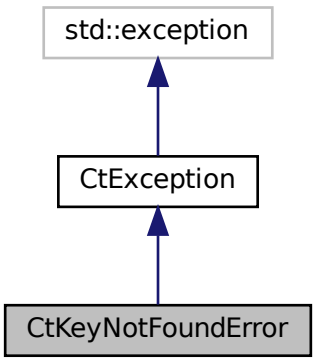
- include/core/exceptions/CtFileExceptions.hpp

8.12 CtKeyNotFoundError Class Reference

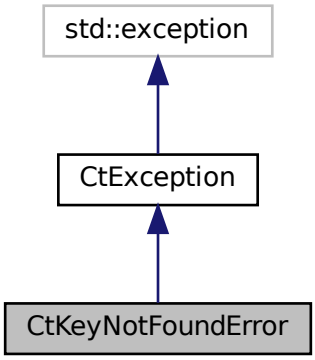
This exception is thrown when a key is not found in a container.

`#include <CtTypeExceptions.hpp>`

Inheritance diagram for CtKeyNotFoundError:



Collaboration diagram for CtKeyNotFoundError:



Public Member Functions

- CtKeyNotFoundError (const CtString &msg)

Additional Inherited Members

8.12.1 Detailed Description

This exception is thrown when a key is not found in a container.

FR-001-001-005 FR-001-001-002 FR-001-001-003

Definition at line 56 of file CtTypeExceptions.hpp.

8.12.2 Constructor & Destructor Documentation

8.12.2.1 CtKeyNotFoundError()

```
CtKeyNotFoundError::CtKeyNotFoundError (
            const CtString & msg )  [inline], [explicit]
```

Definition at line 58 of file CtTypeExceptions.hpp.

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

- include/core/exceptions/CtTypeExceptions.hpp

8.13 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 CtString &componentName="")

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

  Destructor.
- EXPORTED_API void log_debug (const CtString &message)

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

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

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

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

  Log a message with critical log level.

Private Member Functions

- void log (CtLogger::Level level, const CtString &message)

  Log a message with the specified log level.

Static Private Member Functions

- static const CtString levelToString (CtLogger::Level level)

  Given the logger output level in enum CtLogger::Level format this method returns it in a string format.
- static const CtString generateLoggerMsg (CtLogger::Level level, const CtString &component_name, const CtString &message)

  This method generates the message to be printed via logger.

Private Attributes

- CtMutex m_mtx_control
- CtLogger::Level m_level
- CtString m_componentName

8.13.1 Detailed Description

A simple logger with log levels and timestamp.

The CtLogger class provides a mechanism for logging messages with different log levels. The log levels are DEBUG, INFO, WARNING, ERROR, and CRITICAL and can be used to filter messages. The logger also provides a timestamp for each message. It is thread-safe and can be used in multi-threaded environments.

Definition at line 51 of file CtLogger.hpp.

8.13.2 Member Enumeration Documentation

8.13.2.1 Level

```
enum CtLogger::Level  [strong]
```

Enum representing log levels.

FR-004-002-003

Enumerator

DEBUG	
INFO	
WARNING	
ERROR	
CRITICAL	

Definition at line 58 of file CtLogger.hpp.

8.13.3 Constructor & Destructor Documentation

8.13.3.1 CtLogger()

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

Constructs a CtLogger with a component name.

FR-004-002-001

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.

Definition at line 34 of file CtLogger.cpp.

8.13.3.2 ~CtLogger()

```
CtLogger::~CtLogger ( )
```

Destructor.

FR-004-002-002

Definition at line 37 of file CtLogger.cpp.

8.13.4 Member Function Documentation

8.13.4.1 generateLoggerMsg()

```
const CtString CtLogger::generateLoggerMsg (
            CtLogger::Level level,
            const CtString & component_name,
            const CtString & message )  [static], [private]
```

This method generates the message to be printed via logger.

FR-004-002-005

Parameters

level	The level of the message.
component_name	The component's name.
message	The message.

Returns

const CtString The generated message to be printed via logger.

Definition at line 68 of file CtLogger.cpp.

8.13.4.2 levelToString()

```
const CtString CtLogger::levelToString (
            CtLogger::Level level )  [static], [private]
```

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

FR-004-002-005

Parameters

level	The level in enum CtLogger::Level format.

Returns

CtString The level in string format.

Definition at line 77 of file CtLogger.cpp.

8.13.4.3 log()

```
void CtLogger::log (
            CtLogger::Level level,
            const CtString & message )  [private]
```

Log a message with the specified log level.

FR-004-002-004 FR-004-002-005

Parameters

level	The log level.
componentName	The name of the component or module.
message	The log message.

Definition at line 60 of file CtLogger.cpp.

8.13.4.4 log_critical()

```
void CtLogger::log_critical (
            const CtString & message )
```

Log a message with critical log level.

FR-004-002-004

Parameters

message	The log message.

Definition at line 56 of file CtLogger.cpp.

8.13.4.5 log_debug()

```
void CtLogger::log_debug (
            const CtString & message )
```

Log a message with debug log level.

FR-004-002-004

Parameters

message	The log message.

Definition at line 40 of file CtLogger.cpp.

8.13.4.6 log_error()

```
void CtLogger::log_error (
            const CtString & message )
```

Log a message with error log level.

FR-004-002-004

Parameters

message	The log message.

Definition at line 52 of file CtLogger.cpp.

8.13.4.7 log_info()

```
void CtLogger::log_info (
            const CtString & message )
```

Log a message with info log level.

FR-004-002-004

Parameters

message	The log message.

Definition at line 44 of file CtLogger.cpp.

8.13.4.8 log_warning()

```
void CtLogger::log_warning (
            const CtString & message )
```

Log a message with warning log level.

FR-004-002-004

Parameters

message	The log message.

Definition at line 48 of file CtLogger.cpp.

8.13.5 Member Data Documentation

8.13.5.1 m_componentName

```
CtString CtLogger::m_componentName  [private]
```

Component name.

Definition at line 160 of file CtLogger.hpp.

8.13.5.2 m_level

```
CtLogger::Level CtLogger::m_level  [private]
```

Level of message logging.

Definition at line 159 of file CtLogger.hpp.

8.13.5.3 m_mtx_control

```
CtMutex CtLogger::m_mtx_control  [private]
```

Mutex for controlling access to shared resources.

Definition at line 158 of file CtLogger.hpp.

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

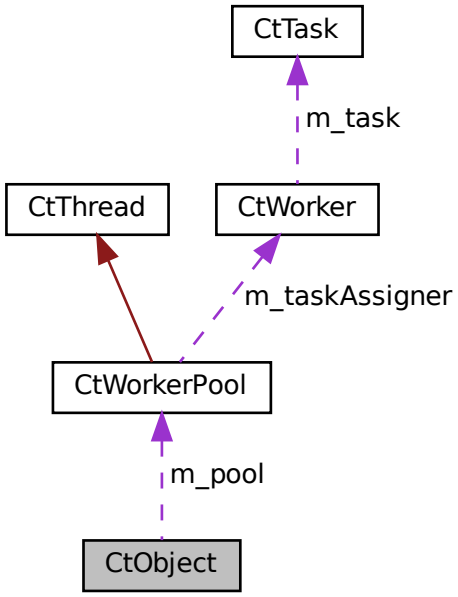
- include/utils/CtLogger.hpp
- src/utils/CtLogger.cpp

8.14 CtObject Class Reference

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

```
#include <CtObject.hpp>
```

Collaboration diagram for CtObject:



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 holds current thread waiting for all the pending events of this object to finish.
- 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.

Private Member Functions

- EXPORTED_API CtBool hasEvent (CtUInt32 p_eventCode)

  This methods checks if a specific event code is already registered in this object.

Private Attributes

- CtMutex m_mtx_control
- CtVector< CtUInt32 > m_events
- CtMultiMap< CtUInt32, CtTask > m_triggers
- CtWorkerPool m_pool

8.14.1 Detailed Description

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

FR-004-003-001

The CtObject class provides a mechanism for connecting events with functions that should be triggered. This class is thread-safe and can be used in multi-threaded environments.

```
triggerEvent(100);
connectEvent(obj, 100, [](){});
connectEvent(obj, 100, [](){});
```

Definition at line 59 of file CtObject.hpp.

8.14.2 Constructor & Destructor Documentation

8.14.2.1 CtObject()

```
CtObject::CtObject ( )  [protected]
```

The constructor of the CtObject class.

FR-004-003-002

Definition at line 36 of file CtObject.cpp.

8.14.2.2 ~CtObject()

```
CtObject::~CtObject ( )  [protected]
```

The destructor of the CtObject class.

FR-004-003-003 FR-004-003-009

Definition at line 39 of file CtObject.cpp.

8.14.3 Member Function Documentation

8.14.3.1 connectEvent() [1/6]

```
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.

FR-004-003-006 FR-004-003-008

Parameters

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

Returns

void

Definition at line 43 of file CtObject.cpp.

8.14.3.2 connectEvent() [2/6]

```
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.

FR-004-003-006 FR-004-003-008

Template Parameters

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

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.

Returns

void

8.14.3.3 connectEvent() [3/6]

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

```
            CtObject * p_obj,
            CtUInt32 p_eventCode,
            F && func,
            FArgs &&...  fargs )
```

Definition at line 196 of file CtObject.hpp.

8.14.3.4 connectEvent() [4/6]

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

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

FR-004-003-006 FR-004-003-008

Parameters

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

Returns

void

Definition at line 51 of file CtObject.cpp.

8.14.3.5 connectEvent() [5/6]

```
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.

FR-004-003-006 FR-004-003-008

Template Parameters

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

Parameters

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

Returns

void

8.14.3.6 connectEvent() [6/6]

```
template<typename F , typename...  FArgs>
void CtObject::connectEvent (
            CtUInt32 p_eventCode,
            F && func,
            FArgs &&...  fargs )
```

Definition at line 203 of file CtObject.hpp.

8.14.3.7 hasEvent()

```
CtBool CtObject::hasEvent (
            CtUInt32 p_eventCode )  [private]
```

This methods checks if a specific event code is already registered in this object.

FR-004-003-005 FR-004-003-008

Parameters

p_eventCode	The event code to be checked.

Returns

CtBool True if the event code is registered, CT_FALSE otherwise.

Definition at line 81 of file CtObject.cpp.

8.14.3.8 registerEvent()

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

This event registers a specific event code.

FR-004-003-004 FR-004-003-005

Parameters

p_eventCode	The event code to be registered.

Returns

void

Definition at line 73 of file CtObject.cpp.

8.14.3.9 triggerEvent()

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

This method triggers a specific event code.

FR-004-003-007 FR-004-003-008

Parameters

p_eventCode	The event code to be triggered.

Returns

void

Definition at line 59 of file CtObject.cpp.

8.14.3.10 waitPendingEvents()

```
void CtObject::waitPendingEvents ( )
```

This method holds current thread waiting for all the pending events of this object to finish.

FR-004-003-010

Returns

void

Definition at line 47 of file CtObject.cpp.

8.14.4 Member Data Documentation

8.14.4.1 m_events

```
CtVector<CtUInt32> CtObject::m_events  [private]
```

This vector contains all the registered event codes.

Definition at line 190 of file CtObject.hpp.

8.14.4.2 m_mtx_control

```
CtMutex CtObject::m_mtx_control  [private]
```

Mutex for controlling access to shared resources.

Definition at line 189 of file CtObject.hpp.

8.14.4.3 m_pool

```
CtWorkerPool CtObject::m_pool  [private]
```

This CtWorkerPool executes the triggered tasks.

Definition at line 192 of file CtObject.hpp.

8.14.4.4 m_triggers

```
CtMultiMap<CtUInt32, CtTask> CtObject::m_triggers  [private]
```

This map represents a list of tasks that should be triggered for each event code.

Definition at line 191 of file CtObject.hpp.

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

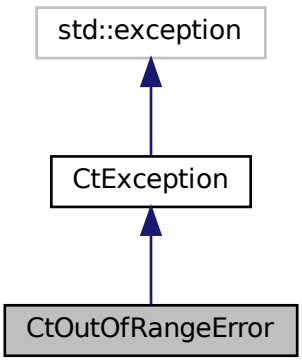
- include/utils/CtObject.hpp
- src/utils/CtObject.cpp

8.15 CtOutOfRangeError Class Reference

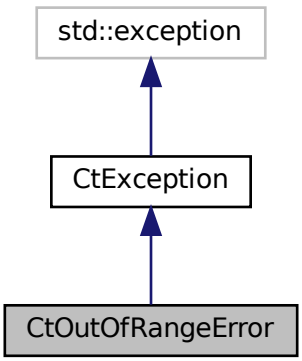
This exception is thrown when an index is out of bounds.

`#include <CtTypeExceptions.hpp>`

Inheritance diagram for CtOutOfRangeError:



Collaboration diagram for CtOutOfRangeError:



Public Member Functions

- CtOutOfRangeError (const CtString &msg)

Additional Inherited Members

8.15.1 Detailed Description

This exception is thrown when an index is out of bounds.

FR-001-001-006 FR-001-001-002 FR-001-001-003

Definition at line 68 of file CtTypeExceptions.hpp.

8.15.2 Constructor & Destructor Documentation

8.15.2.1 CtOutOfRangeError()

```
CtOutOfRangeError::CtOutOfRangeError (
            const CtString & msg )  [inline], [explicit]
```

Definition at line 70 of file CtTypeExceptions.hpp.

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

- include/core/exceptions/CtTypeExceptions.hpp

8.16 CtRawData Class Reference

Struct describing raw data buffer.

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

Public Member Functions

- EXPORTED_API CtRawData (CtUInt32 p_size=CT_BUFFER_SIZE)

  CtRawData constructor.
- EXPORTED_API CtRawData (CtRawData &p_data)

  CtRawData copy constructor.
- virtual EXPORTED_API ~CtRawData ()

  Destructor.
- EXPORTED_API void setNextByte (CtUInt8 p_data)

  Sets the next byte of the buffer. It also raises the size of the buffer. If the buffer is full an exception will be thrown - CtOutOfRangeError()
- EXPORTED_API void setNextBytes (CtUInt8 *p_data, CtUInt32 p_size)

  Sets the next byte of the buffer. It also raises the size of the buffer. If the buffer is full an exception will be thrown - CtOutOfRangeError()
- EXPORTED_API CtUInt8 * getNLastBytes (CtUInt32 p_num)

This method returns a pointer to the last N bytes of the buffer. If the number of bytes is greater than the buffer size an exception will be thrown - CtOutOfRangeError()

- EXPORTED_API void removeNLastBytes (CtUInt32 p_num)

  This method removes the last N bytes of the buffer. It also reduces the size of the buffer. If the number of bytes is greater than the buffer size an exception will be thrown - CtOutOfRangeError()
- EXPORTED_API CtUInt32 size ()

  The actual size of the buffer.
- EXPORTED_API CtUInt32 maxSize ()

  The max size of the buffer.
- EXPORTED_API CtUInt8 * get ()

  This method returns a pointer to the buffer data.
- EXPORTED_API void clone (const CtUInt8 *p_data, CtUInt32 p_size)

  This method fills the buffer with the data given in the parameters. This method overwrites the buffer and the actual size. The maximum size of the buffer is preserved. If the size of the data is greater than the buffer size an exception will be thrown - CtOutOfRangeError()
- EXPORTED_API void clone (CtRawData &p_data)

  This method fills the buffer with the data given in the parameters. This method overwrites the buffer and the actual size. The maximum size of the buffer is preserved. If the size of the data is greater than the buffer size an exception will be thrown - CtOutOfRangeError()
- EXPORTED_API void reset ()

  This method resets the buffer to 0 size. The allocated memory is not freed. The actual size of the buffer is set to 0.
- EXPORTED_API CtRawData & operator= (CtRawData &other)

  Assignment operator for CtRawData. Copies the data from a CtRawData to another CtRawData object.

Private Attributes

- CtUInt8 * m_data
- CtUInt32 m_size
- const CtUInt32 m_maxSize

8.16.1 Detailed Description

Struct describing raw data buffer.

FR-001-003-002

The default buffer size is defined as CT_BUFFER_SIZE bytes. The buffer can be set either by another buffer object or given the size of the buffer. The buffer has a prespecified size that can be filled with bytes. It can monitor the size of the buffer that it is currently used and it ensures that the buffer will not overflow. If an overflow occurs an exception will be thrown - CtOutOfRangeError().

```
CtRawData data;
data.nextByte('a');
data.nextByte('b');
data.nextByte('c');
CtUInt8* buffer = data.get(); // returns "abc"
data.removeNLastBytes(1);
buffer = data.get(); // returns "ab"
data.reset(); // resets the buffer
buffer = data.get(); // returns ""
```

Definition at line 120 of file CtTypes.hpp.

8.16.2 Constructor & Destructor Documentation

8.16.2.1 CtRawData() [1/2]

```
CtRawData::CtRawData (
            CtUInt32 p_size = CT_BUFFER_SIZE )  [explicit]
```

CtRawData constructor.

FR-001-003-003 FR-001-003-004 FR-001-003-007

Parameters

p_size	The size of the buffer. The default size is defined as CT_BUFFER_SIZE bytes.

Definition at line 39 of file CtTypes.cpp.

8.16.2.2 CtRawData() [2/2]

```
CtRawData::CtRawData (
            CtRawData & p_data )
```

CtRawData copy constructor.

FR-001-003-003 FR-001-003-005 FR-001-003-007

Parameters

p_data	Another CtRawData object that it is used to init the currently created.

Definition at line 44 of file CtTypes.cpp.

8.16.2.3 ~CtRawData()

```
CtRawData::~CtRawData ( )  [virtual]
```

Destructor.

FR-001-003-006

Definition at line 49 of file CtTypes.cpp.

8.16.3 Member Function Documentation

8.16.3.1 clone() [1/2]

```
void CtRawData::clone (
            const CtUInt8 * p_data,
            CtUInt32 p_size )
```

This method fills the buffer with the data given in the parameters. This method overwrites the buffer and the actual size. The maximum size of the buffer is preserved. If the size of the data is greater than the buffer size an exception will be thrown - CtOutOfRangeError()

FR-001-003-016 FR-001-003-019

Parameters

p_data	A pointer to the data to be cloned.
p_size	The size of the gven buffer.

Returns

void

Definition at line 94 of file CtTypes.cpp.

8.16.3.2 clone() [2/2]

```
void CtRawData::clone (
            CtRawData & p_data )
```

This method fills the buffer with the data given in the parameters. This method overwrites the buffer and the actual size. The maximum size of the buffer is preserved. If the size of the data is greater than the buffer size an exception will be thrown - CtOutOfRangeError()

FR-001-003-015 FR-001-003-019

Parameters

p_data	A CtRawData object to be cloned.

Returns

void

Definition at line 102 of file CtTypes.cpp.

8.16.3.3 get()

```
CtUInt8 * CtRawData::get ( )
```

This method returns a pointer to the buffer data.

FR-001-003-014

Returns

CtUInt8* Pointer to the buffer data.

Definition at line 90 of file CtTypes.cpp.

8.16.3.4 getNLastBytes()

```
CtUInt8 * CtRawData::getNLastBytes (
            CtUInt32 p_num )
```

This method returns a pointer to the last N bytes of the buffer. If the number of bytes is greater than the buffer size an exception will be thrown - CtOutOfRangeError()

FR-001-003-010 FR-001-003-019

Parameters

p_num	Number of bytes to be returned.

Returns

CtUInt8* Pointer to the last N bytes of the buffer.

Definition at line 68 of file CtTypes.cpp.

8.16.3.5 maxSize()

```
CtUInt32 CtRawData::maxSize ( )
```

The max size of the buffer.

FR-001-003-007 FR-001-003-013

Returns

CtUInt32 The max size of the buffer.

Definition at line 86 of file CtTypes.cpp.

8.16.3.6 operator=()

```
CtRawData & CtRawData::operator= (
            CtRawData & other )
```

Assignment operator for CtRawData. Copies the data from a CtRawData to another CtRawData object.

FR-001-003-018

Parameters

other	The CtRawData object to copy.

Returns

CtRawData& Reference to the current CtRawData object.

Definition at line 114 of file CtTypes.cpp.

8.16.3.7 removeNLastBytes()

```
void CtRawData::removeNLastBytes (
            CtUInt32 p_num )
```

This method removes the last N bytes of the buffer. It also reduces the size of the buffer. If the number of bytes is greater than the buffer size an exception will be thrown - CtOutOfRangeError()

FR-001-003-011 FR-001-003-019

Parameters

p_num	Number of bytes to be returned.

Returns

void

Definition at line 75 of file CtTypes.cpp.

8.16.3.8 reset()

```
void CtRawData::reset ( )
```

This method resets the buffer to 0 size. The allocated memory is not freed. The actual size of the buffer is set to 0.

FR-001-003-017

Returns

void

Definition at line 110 of file CtTypes.cpp.

8.16.3.9 setNextByte()

```
void CtRawData::setNextByte (
            CtUInt8 p_data )
```

Sets the next byte of the buffer. It also raises the size of the buffer. If the buffer is full an exception will be thrown - CtOutOfRangeError()

FR-001-003-008 FR-001-003-019

Parameters

p_data	The byte to be added.

Returns

void

Definition at line 53 of file CtTypes.cpp.

8.16.3.10 setNextBytes()

```
void CtRawData::setNextBytes (
            CtUInt8 * p_data,
            CtUInt32 p_size )
```

Sets the next byte of the buffer. It also raises the size of the buffer. If the buffer is full an exception will be thrown - CtOutOfRangeError()

FR-001-003-009 FR-001-003-019

Parameters

p_data	The byte to be added.
p_size	The number of bytes to be added.

Returns

void

Definition at line 60 of file CtTypes.cpp.

8.16.3.11 size()

```
CtUInt32 CtRawData::size ( )
```

The actual size of the buffer.

FR-001-003-007 FR-001-003-012

Returns

CtUInt32 The actual size of the buffer.

Definition at line 82 of file CtTypes.cpp.

8.16.4 Member Data Documentation

8.16.4.1 m_data

```
CtUInt8* CtRawData::m_data  [private]
```

The buffer data.

Definition at line 281 of file CtTypes.hpp.

8.16.4.2 m_maxSize

```
const CtUInt32 CtRawData::m_maxSize  [private]
```

The maximum size of the buffer.

Definition at line 283 of file CtTypes.hpp.

8.16.4.3 m_size

```
CtUInt32 CtRawData::m_size  [private]
```

The actual size of the buffer.

Definition at line 282 of file CtTypes.hpp.

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

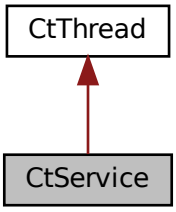
- include/core/CtTypes.hpp
- src/core/CtTypes.cpp

8.17 CtService Class Reference

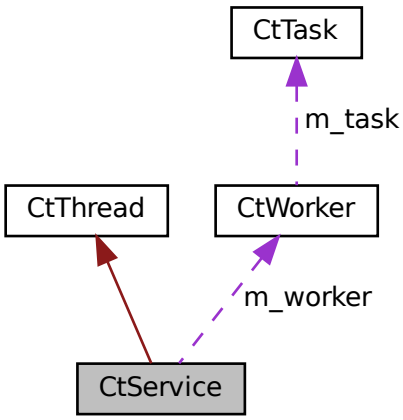
A class representing a service that runs a given task at regular intervals using a worker thread.

`#include <CtService.hpp>`

Inheritance diagram for CtService:



Collaboration diagram for CtService:



Public Member Functions

- EXPORTED_API CtService (CtUInt64 nslots, const CtTask &task)

  Constructor for CtService.
- template<typename F , typename... FArgs>
  EXPORTED_API CtService (CtUInt64 nslots, const 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.

- EXPORTED_API float getIntervalValidity ()

  Get the Interval Validity Factor.

- template<typename F , typename... FArgs>
  CtService (CtUInt64 nslots, const F &&func, FArgs &&... fargs)

Static Public Attributes

- static CtUInt32 m_slot_time = 10

  The time interval for each "slot" in milliseconds.

Private Member Functions

- void loop () override

  Overridden run function from CtThread, representing the main logic of the service.

Private Attributes

- CtWorker m_worker
- CtUInt64 m_nslots
- CtUInt32 m_skip_ctr
- CtUInt32 m_exec_ctr

Additional Inherited Members

8.17.1 Detailed Description

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

FR-005-005-001 FR-005-005-008 FR-005-005-013

The CtService class provides a mechanism for running a task at regular intervals using a worker thread. The service can be configured to run the task immediately or after a certain number of time slots. The service can be stopped and started at any time. The service is thread-safe and can be used in multi-threaded environments.

```
// create a service that runs a task every 1000 milliseconds
CtService service(1000, [](){ std::cout « "Hello from service!" « std::endl; });
service.runService();
// do something else
service.stopService();
```

Definition at line 64 of file CtService.hpp.

8.17.2 Constructor & Destructor Documentation

8.17.2.1 CtService() [1/3]

```
CtService::CtService (
            CtUInt64 nslots,
            const CtTask & task )
```

Constructor for CtService.

FR-005-005-002

Parameters

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

Definition at line 36 of file CtService.cpp.

8.17.2.2 CtService() [2/3]

```
template<typename F , typename...  FArgs>
EXPORTED_API CtService::CtService (
            CtUInt64 nslots,
            const F && func,
            FArgs &&...  fargs )
```

Constructor for CtService.

FR-005-005-002

Parameters

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

8.17.2.3 ~CtService()

```
CtService::~CtService ( )
```

Destructor for CtService.

FR-005-005-003 FR-005-005-004

Definition at line 40 of file CtService.cpp.

8.17.2.4 CtService() [3/3]

```
template<typename F , typename...  FArgs>
CtService::CtService (
            CtUInt64 nslots,
            const F && func,
            FArgs &&...  fargs )
```

Definition at line 160 of file CtService.hpp.

8.17.3 Member Function Documentation

8.17.3.1 getIntervalValidity()

```
float CtService::getIntervalValidity ( )
```

Get the Interval Validity Factor.

FR-005-005-011

This method returns a factor that represents the validity of the interval. If the factor is 1, the task is executed at the correct interval. If the factor is less than 1, the task is executed less frequently than expected. The closer the factor is to 1, the more frequently the task is executed.

Returns

float The interval validity factor.

Definition at line 57 of file CtService.cpp.

8.17.3.2 loop()

```
void CtService::loop ( )  [override], [private], [virtual]
```

Overridden run function from CtThread, representing the main logic of the service.

FR-005-005-008 FR-005-005-009 FR-005-005-010

Implements CtThread.

Definition at line 61 of file CtService.cpp.

8.17.3.3 runService()

```
void CtService::runService ( )
```

Run the task provided by the service.

FR-005-005-005 FR-005-005-007

CtServiceError is thrown in case of service is already running.

Definition at line 44 of file CtService.cpp.

8.17.3.4 stopService()

```
void CtService::stopService ( )
```

Stop the task provided by the service.

FR-005-005-006

Definition at line 52 of file CtService.cpp.

8.17.4 Member Data Documentation

8.17.4.1 m_exec_ctr

```
CtUInt32 CtService::m_exec_ctr  [private]
```

Total loop counter.

Definition at line 156 of file CtService.hpp.

8.17.4.2 m_nslots

```
CtUInt64 CtService::m_nslots  [private]
```

The number of slots to wait before rerunning the service.

Definition at line 154 of file CtService.hpp.

8.17.4.3 m_skip_ctr

```
CtUInt32 CtService::m_skip_ctr  [private]
```

Task execution skip counter.

Definition at line 155 of file CtService.hpp.

8.17.4.4 m_slot_time

```
CtUInt32 CtService::m_slot_time = 10  [static]
```

The time interval for each "slot" in milliseconds.

FR-005-005-012

Definition at line 139 of file CtService.hpp.

8.17.4.5 m_worker

```
CtWorker CtService::m_worker  [private]
```

Worker for executing the task.

Definition at line 153 of file CtService.hpp.

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

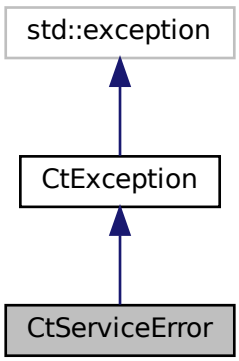
- include/threading/CtService.hpp
- src/threading/CtService.cpp

8.18 CtServiceError Class Reference

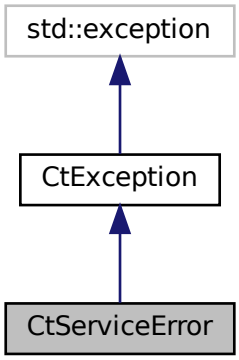
This exception is thrown when a service pool error occurs.

```
#include <CtThreadExceptions.hpp>
```

Inheritance diagram for CtServiceError:



Collaboration diagram for CtServiceError:



Public Member Functions

- CtServiceError (const CtString &msg)

Additional Inherited Members

8.18.1 Detailed Description

This exception is thrown when a service pool error occurs.

FR-001-001-008 FR-001-001-002 FR-001-001-003

Definition at line 56 of file CtThreadExceptions.hpp.

8.18.2 Constructor & Destructor Documentation

8.18.2.1 CtServiceError()

```
CtServiceError::CtServiceError (
            const CtString & msg )  [inline], [explicit]
```

Definition at line 58 of file CtThreadExceptions.hpp.

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

- include/core/exceptions/CtThreadExceptions.hpp

8.19 CtServicePack Struct Reference

Represents a pack containing a task, an ID, and an interval for execution.

8.19.1 Detailed Description

Represents a pack containing a task, an ID, and an interval for execution.

FR-005-006-007

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

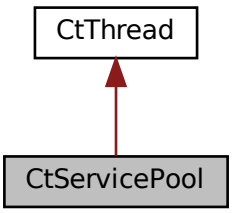
- include/threading/CtServicePool.hpp

8.20 CtServicePool Class Reference

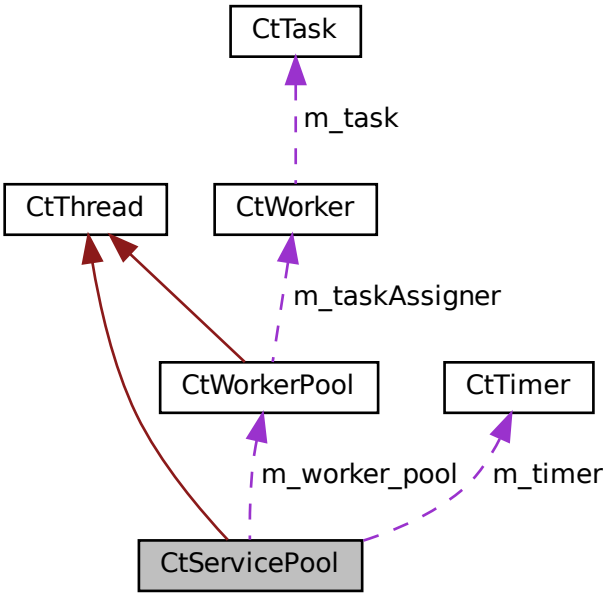
A service pool for managing and executing tasks at specified intervals using a worker pool.

`#include <CtServicePool.hpp>`

Inheritance diagram for CtServicePool:



Collaboration diagram for CtServicePool:



Classes

- struct _CtServicePack

Public Member Functions

- EXPORTED_API CtServicePool (CtUInt32 nworkers)

  Constructor for CtServicePool.
- EXPORTED_API ~CtServicePool ()

  Destructor for CtServicePool.
- EXPORTED_API void addTask (CtUInt32 nslots, const CtString &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, const CtString &id, F &&func, FArgs &&... fargs)

  Add a task to the service pool with a specified interval and an optional ID.
- EXPORTED_API void removeTask (const CtString &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.
- template<typename F , typename... FArgs>
  void addTaskFunc (CtUInt32 nslots, const CtString &id, F &&func, FArgs &&... fargs)

Private Types

- typedef struct CtServicePool::_CtServicePack CtServicePack

Private Member Functions

- void loop () override

  Overridden loop function from CtThread, representing the main thread logic.

Private Attributes

- CtUInt32 m_nworkers
- CtUInt32 m_slot_cnt
- CtVector< CtServicePack > m_tasks
- CtMutex m_mtx_control
- CtWorkerPool m_worker_pool
- CtTimer m_timer
- CtUInt64 m_exec_time

Additional Inherited Members

8.20.1 Detailed Description

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

FR-005-006-001 FR-005-006-008

The CtServicePool class provides a mechanism for managing and executing tasks at specified intervals using a worker pool. CtService::m_slot_time is used to determine the interval at which tasks are executed. The default slot time is 10 ms. You can modify the slot time using the setSlotTime() method. The class uses a worker pool to execute tasks concurrently. The class is thread-safe and can be used in multi-threaded environments.

```
// create a service pool with 4 worker threads
CtServicePool pool(4);
// add tasks to the pool
// add a lambda function
pool.addTask(100, [](){ std::cout « "Hello from worker thread!" « std::endl; });
// add a function with arguments
pool.addTask(100, func, arg1, arg2);
// start the services
pool.startServices();
// stop the services
pool.shutdownServices();
```

Definition at line 70 of file CtServicePool.hpp.

8.20.2 Member Typedef Documentation

8.20.2.1 CtServicePack

```
typedef struct CtServicePool::_CtServicePack CtServicePool::CtServicePack  [private]
```

8.20.3 Constructor & Destructor Documentation

8.20.3.1 CtServicePool()

```
CtServicePool::CtServicePool (
            CtUInt32 nworkers )  [explicit]
```

Constructor for CtServicePool.

FR-005-006-002

Parameters

nworkers	The number of worker threads in the service pool.

Definition at line 34 of file CtServicePool.cpp.

8.20.3.2 ~CtServicePool()

```
CtServicePool::~CtServicePool ( )
```

Destructor for CtServicePool.

FR-005-006-003 FR-005-006-004

Definition at line 39 of file CtServicePool.cpp.

8.20.4 Member Function Documentation

8.20.4.1 addTask()

```
void CtServicePool::addTask (
            CtUInt32 nslots,
            const CtString & id,
            CtTask & task )
```

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

FR-005-006-005

Adding a task to the service pool with a specified interval and an optional ID. The service pool automatically starts when a task is added.

Parameters

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

Definition at line 44 of file CtServicePool.cpp.

8.20.4.2 addTaskFunc() [1/2]

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

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

FR-005-006-005

Adding a task to the service pool with a specified interval and an optional ID. The service pool automatically starts when a task is added.

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.

8.20.4.3 addTaskFunc() [2/2]

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

Definition at line 180 of file CtServicePool.hpp.

8.20.4.4 loop()

```
void CtServicePool::loop ( )  [override], [private], [virtual]
```

Overridden loop function from CtThread, representing the main thread logic.

FR-005-006-011

Implements CtThread.

Definition at line 70 of file CtServicePool.cpp.

8.20.4.5 removeTask()

```
void CtServicePool::removeTask (
            const CtString & id )
```

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

FR-005-006-006

Parameters

id	The ID of the task to be removed.

Definition at line 49 of file CtServicePool.cpp.

8.20.4.6 shutdownServices()

```
void CtServicePool::shutdownServices ( )
```

Shutdown the services provided by the service pool.

FR-005-006-010

Definition at line 65 of file CtServicePool.cpp.

8.20.4.7 startServices()

```
void CtServicePool::startServices ( )
```

Start the services provided by the service pool.

FR-005-006-009

Definition at line 58 of file CtServicePool.cpp.

8.20.5 Member Data Documentation

8.20.5.1 m_exec_time

```
CtUInt64 CtServicePool::m_exec_time  [private]
```

Variable used for time tracking during a loop.

Definition at line 176 of file CtServicePool.hpp.

8.20.5.2 m_mtx_control

```
CtMutex CtServicePool::m_mtx_control  [private]
```

Mutex for controlling access to shared resources.

Definition at line 173 of file CtServicePool.hpp.

8.20.5.3 m_nworkers

```
CtUInt32 CtServicePool::m_nworkers  [private]
```

The number of worker threads in the service pool.

Definition at line 170 of file CtServicePool.hpp.

8.20.5.4 m_slot_cnt

```
CtUInt32 CtServicePool::m_slot_cnt  [private]
```

Counter for the current slot.

Definition at line 171 of file CtServicePool.hpp.

8.20.5.5 m_tasks

```
CtVector<CtServicePack> CtServicePool::m_tasks  [private]
```

Vector of tasks in the service pool.

Definition at line 172 of file CtServicePool.hpp.

8.20.5.6 m_timer

```
CtTimer CtServicePool::m_timer  [private]
```

Timer for tracking time intervals.

Definition at line 175 of file CtServicePool.hpp.

8.20.5.7 m_worker_pool

```
CtWorkerPool CtServicePool::m_worker_pool  [private]
```

Worker pool for executing tasks.

Definition at line 174 of file CtServicePool.hpp.

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

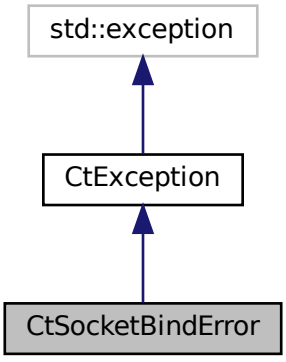
- include/threading/CtServicePool.hpp
- src/threading/CtServicePool.cpp

8.21 CtSocketBindError Class Reference

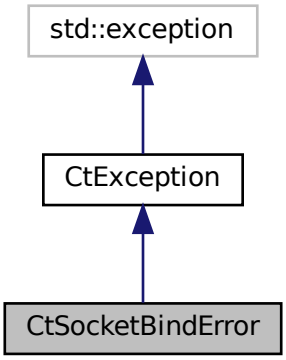
This exception is thrown when a socket bind error occurs.

```
#include <CtNetworkExceptions.hpp>
```

Inheritance diagram for CtSocketBindError:



Collaboration diagram for CtSocketBindError:



Public Member Functions

- CtSocketBindError (const CtString &msg)

Additional Inherited Members

8.21.1 Detailed Description

This exception is thrown when a socket bind error occurs.

FR-001-001-011 FR-001-001-002 FR-001-001-003

Definition at line 56 of file CtNetworkExceptions.hpp.

8.21.2 Constructor & Destructor Documentation

8.21.2.1 CtSocketBindError()

```
CtSocketBindError::CtSocketBindError (
            const CtString & msg )  [inline], [explicit]
```

Definition at line 58 of file CtNetworkExceptions.hpp.

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

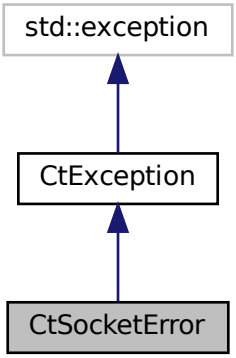
- include/core/exceptions/CtNetworkExceptions.hpp

8.22 CtSocketError Class Reference

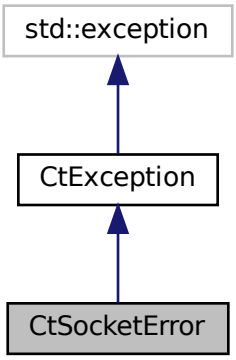
This exception is thrown when a socket error occurs.

`#include <CtNetworkExceptions.hpp>`

Inheritance diagram for CtSocketError:



Collaboration diagram for CtSocketError:



Public Member Functions

- CtSocketError (const CtString &msg)

Additional Inherited Members

8.22.1 Detailed Description

This exception is thrown when a socket error occurs.

FR-001-001-010 FR-001-001-002 FR-001-001-003

Definition at line 44 of file CtNetworkExceptions.hpp.

8.22.2 Constructor & Destructor Documentation

8.22.2.1 CtSocketError()

```
CtSocketError::CtSocketError (
            const CtString & msg )  [inline], [explicit]
```

Definition at line 46 of file CtNetworkExceptions.hpp.

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

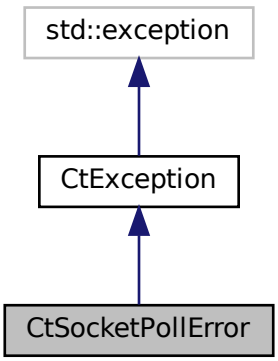
- include/core/exceptions/CtNetworkExceptions.hpp

8.23 CtSocketPollError Class Reference

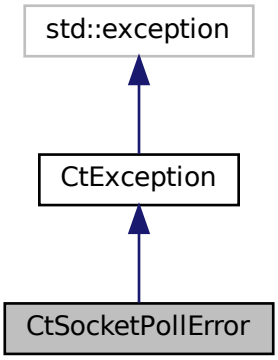
This exception is thrown when a socket listen error occurs.

```
#include <CtNetworkExceptions.hpp>
```

Inheritance diagram for CtSocketPollError:



Collaboration diagram for CtSocketPollError:



Public Member Functions

- CtSocketPollError (const CtString &msg)

Additional Inherited Members

8.23.1 Detailed Description

This exception is thrown when a socket listen error occurs.

FR-001-001-012 FR-001-001-002 FR-001-001-003

Definition at line 68 of file CtNetworkExceptions.hpp.

8.23.2 Constructor & Destructor Documentation

8.23.2.1 CtSocketPollError()

```
CtSocketPollError::CtSocketPollError (
            const CtString & msg )  [inline], [explicit]
```

Definition at line 70 of file CtNetworkExceptions.hpp.

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

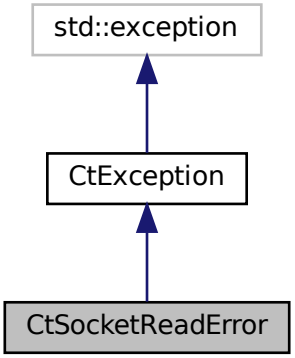
- include/core/exceptions/CtNetworkExceptions.hpp

8.24 CtSocketReadError Class Reference

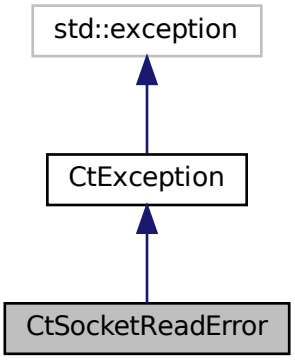
This exception is thrown when a socket accept error occurs.

`#include <CtNetworkExceptions.hpp>`

Inheritance diagram for CtSocketReadError:



Collaboration diagram for CtSocketReadError:



Public Member Functions

- CtSocketReadError (const CtString &msg)

Additional Inherited Members

8.24.1 Detailed Description

This exception is thrown when a socket accept error occurs.

FR-001-001-013 FR-001-001-002 FR-001-001-003

Definition at line 80 of file CtNetworkExceptions.hpp.

8.24.2 Constructor & Destructor Documentation

8.24.2.1 CtSocketReadError()

```
CtSocketReadError::CtSocketReadError (
            const CtString & msg )  [inline], [explicit]
```

Definition at line 82 of file CtNetworkExceptions.hpp.

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

- include/core/exceptions/CtNetworkExceptions.hpp

8.25 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 CtString &p_interfaceName, CtUInt16 p_port)
  Set the socket for subscribing.
- EXPORTED_API void setPub (CtUInt16 p_port, const CtString &p_addr="0.0.0.0")
  Set the socket for publishing.
- EXPORTED_API CtBool pollRead ()
  Check if there is data available to read.
- EXPORTED_API CtBool pollWrite ()
  Check if data can be written to the fd.
- EXPORTED_API void send (CtUInt8 *p_data, CtUInt32 p_size)
  Send data over the socket.
- EXPORTED_API void send (CtRawData &p_message)
  Send data over the socket.
- EXPORTED_API void receive (CtUInt8 *p_data, CtUInt32 p_size, CtNetAddress *p_client=nullptr)
  Receive data from the socket.
- EXPORTED_API void receive (CtRawData *p_message, CtNetAddress *p_clientAddress=nullptr)
  Receive data from the socket.

Private Attributes

- int m_addrType
- int m_socket
- CtUInt16 m_port
- CtString m_addr
- struct pollfd m_pollin_sockets [1]
- struct pollfd m_pollout_sockets [1]
- sockaddr_in m_pubAddress
- sockaddr_in m_subAddress

8.25.1 Detailed Description

A class representing a UDP socket wrapper.

This class provides an interface for creating and managing UDP sockets. It can be used for both subscribing and publishing data. CtSocketHelpers::socketTimeout can be used to set the timeout for polling operations. By default, the timeout is set to 0 which means that the poll operation will return immediately. If the timeout is set to -1, the poll operation will block indefinitely. If set to a positive value, the poll operation will block for that amount of time.

Example subscriber:

```
// create a UDP socket
CtSocketUdp socket;
// set the socket for subscribing
socket.setSub("lo", 1234);
// run a loop to receive messages
while (CT_TRUE) {
     if (socket.pollRead()) {
         CtRawData message;
         socket.receive(&message);
         std::cout « "Received message: " « message.get() « std::endl;
     }
}
```

Example publisher:

```
// create a UDP socket
CtSocketUdp socket;
// set the socket for publishing
socket.setPub(1234, "127.0.0.1");
// send a message
CtRawData message("Hello, World!");
socket.send(message);
```

Definition at line 82 of file CtSocketUdp.hpp.

8.25.2 Constructor & Destructor Documentation

8.25.2.1 CtSocketUdp()

```
CtSocketUdp::CtSocketUdp ( )
```

Constructor for CtSocketUdp.

FR-006-001-001 FR-006-001-003

Definition at line 34 of file CtSocketUdp.cpp.

8.25.2.2 ~CtSocketUdp()

```
CtSocketUdp::~CtSocketUdp ( )
```

Destructor for CtSocketUdp.

FR-006-001-002

Definition at line 48 of file CtSocketUdp.cpp.

8.25.3 Member Function Documentation

8.25.3.1 pollRead()

```
CtBool CtSocketUdp::pollRead ( )
```

Check if there is data available to read.

FR-006-001-006 FR-006-001-008

Returns

True if data is available, CT_FALSE otherwise.

Definition at line 70 of file CtSocketUdp.cpp.

8.25.3.2 pollWrite()

```
CtBool CtSocketUdp::pollWrite ( )
```

Check if data can be written to the fd.

FR-006-001-007 FR-006-001-008

Returns

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

Definition at line 82 of file CtSocketUdp.cpp.

8.25.3.3 receive() [1/2]

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

Receive data from the socket.

FR-006-001-010 FR-006-001-012

Parameters

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

Definition at line 120 of file CtSocketUdp.cpp.

8.25.3.4 receive() [2/2]

```
void CtSocketUdp::receive (
            CtUInt8 * p_data,
            CtUInt32 p_size,
            CtNetAddress * p_client = nullptr )
```

Receive data from the socket.

FR-006-001-010 FR-006-001-012

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).

Definition at line 103 of file CtSocketUdp.cpp.

8.25.3.5 send() [1/2]

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

Send data over the socket.

FR-006-001-009 FR-006-001-011

Parameters

p_message	Struct containing the message to sent.

Definition at line 99 of file CtSocketUdp.cpp.

8.25.3.6 send() [2/2]

```
void CtSocketUdp::send (
```

```
            CtUInt8 * p_data,
            CtUInt32 p_size )
```

Send data over the socket.

FR-006-001-009 FR-006-001-011

Parameters

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

Definition at line 93 of file CtSocketUdp.cpp.

8.25.3.7 setPub()

```
void CtSocketUdp::setPub (
            CtUInt16 p_port,
            const CtString & p_addr = "0.0.0.0" )
```

Set the socket for publishing.

FR-006-001-005

Parameters

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

Definition at line 63 of file CtSocketUdp.cpp.

8.25.3.8 setSub()

```
void CtSocketUdp::setSub (
            const CtString & p_interfaceName,
            CtUInt16 p_port )
```

Set the socket for subscribing.

FR-006-001-004

Parameters

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

Definition at line 52 of file CtSocketUdp.cpp.

8.25.4 Member Data Documentation

8.25.4.1 m_addr

```
CtString CtSocketUdp::m_addr  [private]
```

The address associated with the socket.

Definition at line 189 of file CtSocketUdp.hpp.

8.25.4.2 m_addrType

```
int CtSocketUdp::m_addrType  [private]
```

The socket domain (IPv4 or IPv6).

Definition at line 186 of file CtSocketUdp.hpp.

8.25.4.3 m_pollin_sockets

```
struct pollfd CtSocketUdp::m_pollin_sockets[1]  [private]
```

Array for polling-in file descriptors.

Definition at line 189 of file CtSocketUdp.hpp.

8.25.4.4 m_pollout_sockets

```
struct pollfd CtSocketUdp::m_pollout_sockets[1]  [private]
```

Array for polling-out file descriptors.

Definition at line 189 of file CtSocketUdp.hpp.

8.25.4.5 m_port

```
CtUInt16 CtSocketUdp::m_port  [private]
```

The port associated with the socket.

Definition at line 188 of file CtSocketUdp.hpp.

8.25.4.6 m_pubAddress

```
sockaddr_in CtSocketUdp::m_pubAddress  [private]
```

The address for publishing data.

Definition at line 192 of file CtSocketUdp.hpp.

8.25.4.7 m_socket

```
int CtSocketUdp::m_socket  [private]
```

The socket descriptor.

Definition at line 187 of file CtSocketUdp.hpp.

8.25.4.8 m_subAddress

```
sockaddr_in CtSocketUdp::m_subAddress  [private]
```

The address for subscribing to data.

Definition at line 193 of file CtSocketUdp.hpp.

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

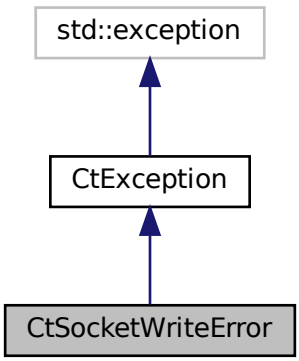
- include/networking/CtSocketUdp.hpp
- src/networking/CtSocketUdp.cpp

8.26 CtSocketWriteError Class Reference

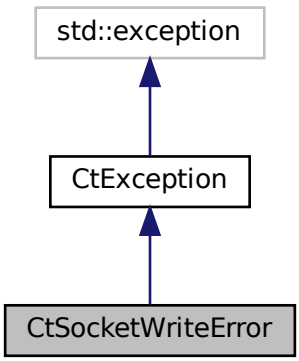
This exception is thrown when a socket connect error occurs.

`#include <CtNetworkExceptions.hpp>`

Inheritance diagram for CtSocketWriteError:



Collaboration diagram for CtSocketWriteError:



Public Member Functions

- CtSocketWriteError (const CtString &msg)

Additional Inherited Members

8.26.1 Detailed Description

This exception is thrown when a socket connect error occurs.

FR-001-001-014 FR-001-001-002 FR-001-001-003

Definition at line 92 of file CtNetworkExceptions.hpp.

8.26.2 Constructor & Destructor Documentation

8.26.2.1 CtSocketWriteError()

```
CtSocketWriteError::CtSocketWriteError (
            const CtString & msg )  [inline], [explicit]
```

Definition at line 94 of file CtNetworkExceptions.hpp.

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

- include/core/exceptions/CtNetworkExceptions.hpp

8.27 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 (const F &&func, FArgs &&... fargs)

  Set the main task function. The task function can also have arguments.
- template<typename C , typename... CArgs>
  EXPORTED_API void setCallbackFunc (const C &&callback, CArgs &&... cargs)

  Set the callback function. The callback function can also have arguments.
- 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 (const F &&func, FArgs &&... fargs)
- template<typename C , typename... CArgs>
  void setCallbackFunc (const C &&callback, CArgs &&... cargs)

Private Attributes

- std::function< void()> m_task
- std::function< void()> m_callback

8.27.1 Detailed Description

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

The task function is the main function that will be executed. The callback function is the function that will be executed after the task function. The task and callback functions can have arguments. This method can be used to organise a specific functionality and post process of it in a single object.

```
CtTask task;
// Set the task function to a lambda function that prints the sum of two integers.
task.setTaskFunc([](int a, int b) {
  std::cout « "Task function: " « a + b « std::endl;
}, 1, 2);
task.setCallbackFunc([](int a, int b) {
  std::cout « "Callback function: " « a - b « std::endl;
}, 1, 2);
// Set the task function to a function with arguments.
task.setTaskFunc(func, arg1, arg2);
task.setCallbackFunc(callbackFunc, arg1, arg2);
```

Definition at line 61 of file CtTask.hpp.

8.27.2 Constructor & Destructor Documentation

8.27.2.1 CtTask() [1/2]

```
CtTask::CtTask ( )  [explicit]
```

Default constructor for CtTask. Initializes task and callback with empty lambda functions.

FR-005-001-002

Definition at line 34 of file CtTask.cpp.

8.27.2.2 CtTask() [2/2]

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

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

FR-005-001-001 FR-005-001-002

Parameters

other	The CtTask object to copy.

Definition at line 37 of file CtTask.cpp.

8.27.2.3 ~CtTask()

```
CtTask::~CtTask ( )
```

Destructor for CtTask.

FR-005-001-003

Definition at line 40 of file CtTask.cpp.

8.27.3 Member Function Documentation

8.27.3.1 getCallbackFunc()

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

Get the callback function.

FR-005-001-001 FR-005-001-005

Returns

The callback function.

Definition at line 47 of file CtTask.cpp.

8.27.3.2 getTaskFunc()

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

Get the main task function.

FR-005-001-001 FR-005-001-004

Returns

The main task function.

Definition at line 43 of file CtTask.cpp.

8.27.3.3 operator=()

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

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

FR-005-001-001 FR-005-001-008

Parameters

other	The CtTask object to copy.

Returns

CtTask& Reference to the current CtTask object.

Definition at line 51 of file CtTask.cpp.

8.27.3.4 setCallbackFunc() [1/2]

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

Set the callback function. The callback function can also have arguments.

FR-005-001-001 FR-005-001-007

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.

Returns

void

8.27.3.5 setCallbackFunc() [2/2]

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

Definition at line 169 of file CtTask.hpp.

8.27.3.6 setTaskFunc() [1/2]

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

Set the main task function. The task function can also have arguments.

FR-005-001-001 FR-005-001-006

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.

Returns

void

8.27.3.7 setTaskFunc() [2/2]

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

Definition at line 164 of file CtTask.hpp.

8.27.4 Member Data Documentation

8.27.4.1 m_callback

```
std::function<void()> CtTask::m_callback  [private]
```

The callback function

Definition at line 160 of file CtTask.hpp.

8.27.4.2 m_task

```
std::function<void()> CtTask::m_task  [private]
```

The main task function

Definition at line 159 of file CtTask.hpp.

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

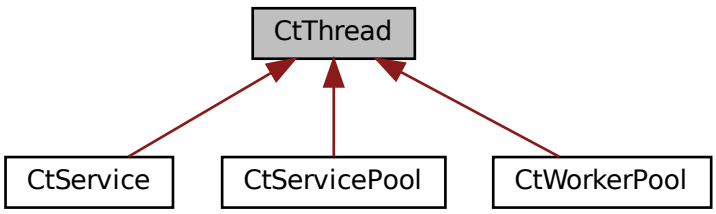
- include/threading/CtTask.hpp
- src/threading/CtTask.cpp

8.28 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 (CtUInt64 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 CtBool isRunning ()
  Check if the thread is currently running.
- EXPORTED_API void start ()
  Start the thread.
- EXPORTED_API void stop ()
  Stop the thread.
- virtual 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 (CtBool running)
  Set the running state of the thread.

Private Member Functions

- void run ()

  Run method executes main loop of each thread.

Private Attributes

- CtAtomic< CtBool > m_running
- std::thread m_thread

8.28.1 Detailed Description

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

FR-005-002-001

The CtThread class provides a simple interface for creating and managing threads in C++. The class is thread-safe and can be used in multi-threaded environments. It is intended to be used as a base class for creating custom thread classes.

Definition at line 51 of file CtThread.hpp.

8.28.2 Constructor & Destructor Documentation

8.28.2.1 CtThread()

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

Constructor for CtThread.

FR-005-002-003

Definition at line 36 of file CtThread.cpp.

8.28.2.2 ~CtThread()

```
CtThread::~CtThread ( )  [protected], [virtual]
```

Virtual destructor for CtThread.

FR-005-002-011

Definition at line 39 of file CtThread.cpp.

8.28.3 Member Function Documentation

8.28.3.1 isRunning()

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

Check if the thread is currently running.

FR-005-002-004 FR-005-002-005

Returns

True if the thread is running, CT_FALSE otherwise.

Definition at line 70 of file CtThread.cpp.

8.28.3.2 join()

```
void CtThread::join ( )  [protected], [virtual]
```

Join the thread, waiting for it to finish.

FR-005-002-012

Reimplemented in CtWorkerPool.

Definition at line 64 of file CtThread.cpp.

8.28.3.3 loop()

```
virtual EXPORTED_API void CtThread::loop ( )  [protected], [pure virtual]
```

Virtual function to be overridden by derived classes. Represents the main functionality of the thread.

FR-005-002-002

Implemented in CtWorkerPool, CtServicePool, and CtService.

8.28.3.4 run()

```
void CtThread::run ( )  [private]
```

Run method executes main loop of each thread.

Definition at line 43 of file CtThread.cpp.

8.28.3.5 setRunning()

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

Set the running state of the thread.

FR-005-002-006

Parameters

running	The running state to set.

Definition at line 74 of file CtThread.cpp.

8.28.3.6 sleepFor()

```
void CtThread::sleepFor (
            CtUInt64 time )  [static]
```

Make the thread sleep for a specified duration in milliseconds.

FR-005-002-013

Parameters

time	Duration to sleep in milliseconds.

Definition at line 78 of file CtThread.cpp.

8.28.3.7 start()

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

Start the thread.

Exceptions

CtThreadError	if the thread is already running.

FR-005-002-007 FR-005-002-008

Definition at line 49 of file CtThread.cpp.

8.28.3.8 stop()

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

Stop the thread.

FR-005-002-009 FR-005-002-010

Definition at line 59 of file CtThread.cpp.

8.28.4 Member Data Documentation

8.28.4.1 m_running

```
CtAtomic<CtBool> CtThread::m_running  [private]
```

Atomic flag indicating whether the thread is running.

Definition at line 139 of file CtThread.hpp.

8.28.4.2 m_thread

```
std::thread CtThread::m_thread  [private]
```

The underlying thread object.

Definition at line 140 of file CtThread.hpp.

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

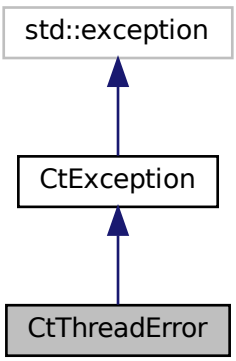
- include/threading/CtThread.hpp
- src/threading/CtThread.cpp

8.29 CtThreadError Class Reference

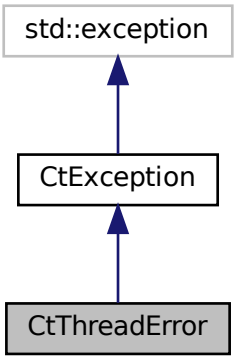
This exception is thrown when a thread error occurs.

```
#include <CtThreadExceptions.hpp>
```

Inheritance diagram for CtThreadError:



Collaboration diagram for CtThreadError:



Public Member Functions

- CtThreadError (const CtString &msg)

Additional Inherited Members

8.29.1 Detailed Description

This exception is thrown when a thread error occurs.

FR-001-001-007 FR-001-001-002 FR-001-001-003

Definition at line 44 of file CtThreadExceptions.hpp.

8.29.2 Constructor & Destructor Documentation

8.29.2.1 CtThreadError()

```
CtThreadError::CtThreadError (
            const CtString & msg )  [inline], [explicit]
```

Definition at line 46 of file CtThreadExceptions.hpp.

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

- include/core/exceptions/CtThreadExceptions.hpp

8.30 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 CtUInt64 toc ()

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

Static Public Member Functions

- static EXPORTED_API CtUInt64 current ()

  Get the current time in milliseconds.

Private Attributes

- CtUInt64 m_reference

8.30.1 Detailed Description

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

The CtTimer class provides a simple interface for measuring elapsed time.

```
CtTimer timer;
timer.tic();
// Do something
CtUInt64 elapsed = timer.toc();
std::cout « "Elapsed time: " « elapsed « " ms" « std::endl;
```

Definition at line 55 of file CtTimer.hpp.

8.30.2 Constructor & Destructor Documentation

8.30.2.1 CtTimer()

```
CtTimer::CtTimer ( )
```

Constructor for CtTimer.

FR-003-001-001

Definition at line 34 of file CtTimer.cpp.

8.30.2.2 ~CtTimer()

```
CtTimer::~CtTimer ( )
```

Destructor for CtTimer.

FR-003-001-002

Definition at line 38 of file CtTimer.cpp.

8.30.3 Member Function Documentation

8.30.3.1 current()

```
CtUInt64 CtTimer::current ( )  [static]
```

Get the current time in milliseconds.

FR-003-001-005

Returns

Current time in milliseconds.

Definition at line 49 of file CtTimer.cpp.

8.30.3.2 tic()

```
void CtTimer::tic ( )
```

Record the current time as a reference point.

FR-003-001-003

Definition at line 41 of file CtTimer.cpp.

8.30.3.3 toc()

```
CtUInt64 CtTimer::toc ( )
```

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

FR-003-001-004

Returns

Elapsed time in milliseconds.

Definition at line 45 of file CtTimer.cpp.

8.30.4 Member Data Documentation

8.30.4.1 m_reference

```
CtUInt64 CtTimer::m_reference  [private]
```

Reference time for measuring elapsed time.

Definition at line 97 of file CtTimer.hpp.

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

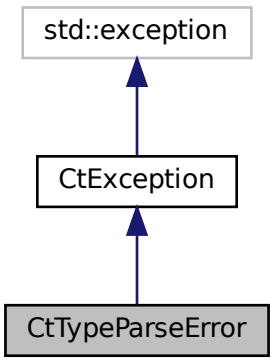
- include/time/CtTimer.hpp
- src/time/CtTimer.cpp

8.31 CtTypeParseError Class Reference

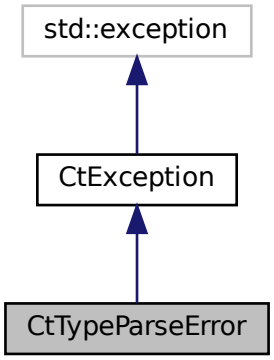
This exception is thrown when a type cannot be parsed.

```
#include <CtTypeExceptions.hpp>
```

Inheritance diagram for CtTypeParseError:



Collaboration diagram for CtTypeParseError:



Public Member Functions

- CtTypeParseError (const CtString &msg)

Additional Inherited Members

8.31.1 Detailed Description

This exception is thrown when a type cannot be parsed.

FR-001-001-004 FR-001-001-002 FR-001-001-003

Definition at line 44 of file CtTypeExceptions.hpp.

8.31.2 Constructor & Destructor Documentation

8.31.2.1 CtTypeParseError()

```
CtTypeParseError::CtTypeParseError (
            const CtString & msg )  [inline], [explicit]
```

Definition at line 46 of file CtTypeExceptions.hpp.

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

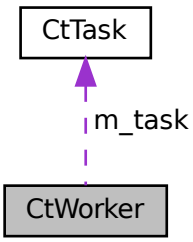
- include/core/exceptions/CtTypeExceptions.hpp

8.32 CtWorker Class Reference

The CtWorker class provides a mechanism for executing tasks asynchronously in a separate thread.

`#include <CtWorker.hpp>`

Collaboration diagram for CtWorker:



Public Member Functions

- EXPORTED_API CtWorker ()

  Constructor for CtWorker.
- EXPORTED_API ~CtWorker ()

  Destructor for CtWorker.
- EXPORTED_API CtBool isRunning ()

  Returns CT_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 (const CtTask &task, std::function< void()> callback=[]{})

  Set a task for the worker to execute.
- template<typename F , typename... FArgs>
  EXPORTED_API void setTaskFunc (const F &&func, FArgs &&... fargs)

  Set a task function for the worker to execute.
- template<typename F , typename... FArgs>
  void setTaskFunc (const F &&func, FArgs &&... fargs)

Private Member Functions

- void alreadyRunningCheck ()

  Helper function that checks if the CtWorker is already running and returns an exception.
- void setRunning (CtBool running)

  Helper function set the m_running flag.

Private Attributes

- CtTask m_task
- CtAtomic< CtBool > m_running
- std::thread m_thread
- std::function< void()> m_callback

8.32.1 Detailed Description

The CtWorker class provides a mechanism for executing tasks asynchronously in a separate thread.

FR-005-003-001

```
CtWorker worker;
// add a lambda function to the worker
worker.setTask([](){ std::cout « "Hello from worker thread!" « std::endl; });
// or add a task
worker.setTask(task);
// or add a function with arguments
worker.setTaskFunc(func, arg1, arg2);
// run the task
worker.runTask();
// wait for the task to complete
worker.joinTask();
```

Definition at line 64 of file CtWorker.hpp.

8.32.2 Constructor & Destructor Documentation

8.32.2.1 CtWorker()

```
CtWorker::CtWorker ( )  [explicit]
```

Constructor for CtWorker.

FR-005-003-002

Definition at line 34 of file CtWorker.cpp.

8.32.2.2 ~CtWorker()

```
CtWorker::~CtWorker ( )
```

Destructor for CtWorker.

FR-005-003-003

Definition at line 37 of file CtWorker.cpp.

8.32.3 Member Function Documentation

8.32.3.1 alreadyRunningCheck()

```
void CtWorker::alreadyRunningCheck ( )  [private]
```

Helper function that checks if the CtWorker is already running and returns an exception.

FR-005-003-011

Definition at line 72 of file CtWorker.cpp.

8.32.3.2 isRunning()

```
CtBool CtWorker::isRunning ( )
```

Returns CT_TRUE if the worker is currently running.

FR-005-003-004

Returns

EXPORTED_API Worker status.

Definition at line 41 of file CtWorker.cpp.

8.32.3.3 joinTask()

```
void CtWorker::joinTask ( )
```

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

FR-005-003-006

Definition at line 66 of file CtWorker.cpp.

8.32.3.4 runTask()

```
void CtWorker::runTask ( )
```

Run the task assigned to the worker.

FR-005-003-010 FR-005-003-011

Definition at line 55 of file CtWorker.cpp.

8.32.3.5 setRunning()

```
void CtWorker::setRunning (
            CtBool running )  [private]
```

Helper function set the m_running flag.

FR-005-003-005

Definition at line 45 of file CtWorker.cpp.

8.32.3.6 setTask()

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

Set a task for the worker to execute.

FR-005-003-007 FR-005-003-008 FR-005-003-009

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.

Definition at line 49 of file CtWorker.cpp.

8.32.3.7 setTaskFunc() [1/2]

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

Set a task function for the worker to execute.

FR-005-003-007 FR-005-003-008

Parameters

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

8.32.3.8 setTaskFunc() [2/2]

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

Definition at line 158 of file CtWorker.hpp.

8.32.4 Member Data Documentation

8.32.4.1 m_callback

```
std::function<void()> CtWorker::m_callback  [private]
```

Callback function to be executed after the task is completed.

Definition at line 154 of file CtWorker.hpp.

8.32.4.2 m_running

```
CtAtomic<CtBool> CtWorker::m_running  [private]
```

Flag indicating if the worker is currently running.

Definition at line 152 of file CtWorker.hpp.

8.32.4.3 m_task

```
CtTask CtWorker::m_task  [private]
```

The task assigned to the worker.

Definition at line 151 of file CtWorker.hpp.

8.32.4.4 m_thread

```
std::thread CtWorker::m_thread  [private]
```

The worker's thread.

Definition at line 153 of file CtWorker.hpp.

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

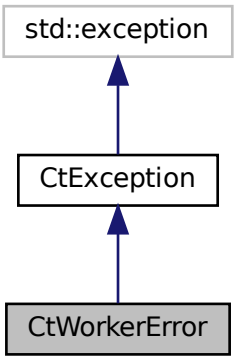
- include/threading/CtWorker.hpp
- src/threading/CtWorker.cpp

8.33 CtWorkerError Class Reference

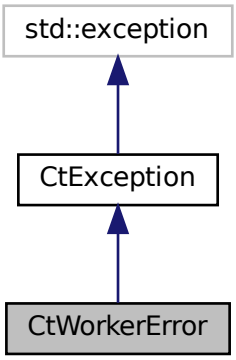
This exception is thrown when a worker error occurs.

```
#include <CtThreadExceptions.hpp>
```

Inheritance diagram for CtWorkerError:



Collaboration diagram for CtWorkerError:



Public Member Functions

- CtWorkerError (const CtString &msg)

Additional Inherited Members

8.33.1 Detailed Description

This exception is thrown when a worker error occurs.

FR-001-001-009 FR-001-001-002 FR-001-001-003

Definition at line 68 of file CtThreadExceptions.hpp.

8.33.2 Constructor & Destructor Documentation

8.33.2.1 CtWorkerError()

```
CtWorkerError::CtWorkerError (
            const CtString & msg )  [inline], [explicit]
```

Definition at line 70 of file CtThreadExceptions.hpp.

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

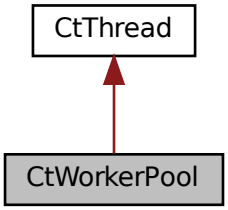
- include/core/exceptions/CtThreadExceptions.hpp

8.34 CtWorkerPool Class Reference

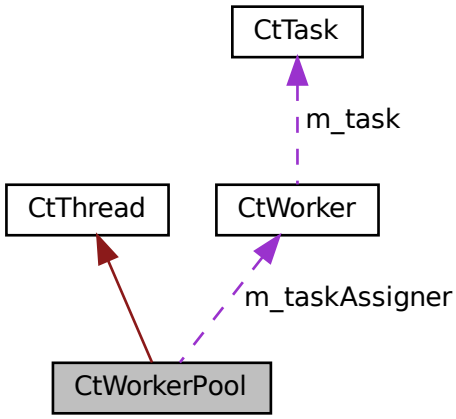
Manages a pool of worker threads for executing tasks concurrently.

`#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 (const CtTask &task)

  Add a task to the worker pool.

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

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

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

Private Member Functions

- void assignTask (CtUInt32 idx)

  Assign a task to a specified worker.
- void free ()

  Free resources and clear the worker pool.
- void loop () override

  Main loop for the worker pool.

Private Attributes

- CtUInt32 m_nworkers
- CtVector< std::unique_ptr< CtWorker > > m_workers
- CtQueue< CtTask > m_tasks
- CtQueue< CtUInt32 > m_available_workers_idxs
- CtMutex m_mtx_control
- CtAtomic< CtUInt32 > m_active_tasks
- CtAtomic< CtUInt32 > m_queued_tasks
- CtWorker m_taskAssigner

Additional Inherited Members

8.34.1 Detailed Description

Manages a pool of worker threads for executing tasks concurrently.

FR-005-004-001 FR-005-004-002

The CtWorkerPool class provides a mechanism for managing a pool of worker threads that can execute tasks concurrently. The class is thread-safe and can be used in multi-threaded environments.

```
// create a pool with 4 worker threads
CtWorkerPool pool(4);
// add tasks to the pool
// add a lambda function
pool.addTask([](){ std::cout « "Hello from worker thread!" « std::endl; });
// add a function with arguments
pool.addTask(func, arg1, arg2);
// add a CtTask object
pool.addTask(task);
// wait for all worker threads to finish
pool.join();
```

Definition at line 69 of file CtWorkerPool.hpp.

8.34.2 Constructor & Destructor Documentation

8.34.2.1 CtWorkerPool()

```
CtWorkerPool::CtWorkerPool (
            CtUInt32 nworkers )  [explicit]
```

Constructor for CtWorkerPool.

FR-005-004-003

Parameters

nworkers	The number of worker threads in the pool.

Definition at line 34 of file CtWorkerPool.cpp.

8.34.2.2 ~CtWorkerPool()

```
CtWorkerPool::~CtWorkerPool ( )
```

Destructor for CtWorkerPool.

FR-005-004-004 FR-005-004-005

Definition at line 40 of file CtWorkerPool.cpp.

8.34.3 Member Function Documentation

8.34.3.1 addTask() [1/3]

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

Add a task to the worker pool.

FR-005-004-006

Parameters

task	The task to be added to the pool.

Definition at line 45 of file CtWorkerPool.cpp.

8.34.3.2 addTask() [2/3]

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

Add a task function to the worker pool.

FR-005-004-006

Parameters

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

8.34.3.3 addTask() [3/3]

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

Definition at line 157 of file CtWorkerPool.hpp.

8.34.3.4 assignTask()

```
void CtWorkerPool::assignTask (
            CtUInt32 idx )  [private]
```

Assign a task to a specified worker.

FR-005-004-008

Parameters

idx	The index of the worker to which the task is assigned.

Returns

True if a task was successfully assigned, CT_FALSE otherwise.

Definition at line 59 of file CtWorkerPool.cpp.

8.34.3.5 free()

```
void CtWorkerPool::free ( )  [private]
```

Free resources and clear the worker pool.

FR-005-004-004

Definition at line 72 of file CtWorkerPool.cpp.

8.34.3.6 join()

```
void CtWorkerPool::join ( )  [override], [virtual]
```

Wait for all worker threads to finish their tasks.

FR-005-004-007

Reimplemented from CtThread.

Definition at line 55 of file CtWorkerPool.cpp.

8.34.3.7 loop()

```
void CtWorkerPool::loop ( )  [override], [private], [virtual]
```

Main loop for the worker pool.

FR-005-004-008 FR-005-004-009

Implements CtThread.

Definition at line 79 of file CtWorkerPool.cpp.

8.34.4 Member Data Documentation

8.34.4.1 m_active_tasks

```
CtAtomic<CtUInt32> CtWorkerPool::m_active_tasks  [private]
```

Number of active tasks that are currently running.

Definition at line 151 of file CtWorkerPool.hpp.

8.34.4.2 m_available_workers_idxs

```
CtQueue<CtUInt32> CtWorkerPool::m_available_workers_idxs  [private]
```

Queue of available worker indices.

Definition at line 149 of file CtWorkerPool.hpp.

8.34.4.3 m_mtx_control

```
CtMutex CtWorkerPool::m_mtx_control  [private]
```

Mutex for controlling access to shared resources.

Definition at line 150 of file CtWorkerPool.hpp.

8.34.4.4 m_nworkers

```
CtUInt32 CtWorkerPool::m_nworkers  [private]
```

Number of worker threads in the pool.

Definition at line 146 of file CtWorkerPool.hpp.

8.34.4.5 m_queued_tasks

```
CtAtomic<CtUInt32> CtWorkerPool::m_queued_tasks  [private]
```

Number of queued tasks.

Definition at line 152 of file CtWorkerPool.hpp.

8.34.4.6 m_taskAssigner

```
CtWorker CtWorkerPool::m_taskAssigner  [private]
```

This worker is a task assigner, assigns active tasks to available workers.

Definition at line 153 of file CtWorkerPool.hpp.

8.34.4.7 m_tasks

```
CtQueue<CtTask> CtWorkerPool::m_tasks  [private]
```

Queue of tasks to be executed.

Definition at line 148 of file CtWorkerPool.hpp.

8.34.4.8 m_workers

```
CtVector<std::unique_ptr<CtWorker> > CtWorkerPool::m_workers  [private]
```

Worker thread instances.

Definition at line 147 of file CtWorkerPool.hpp.

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

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

Chapter 9

File Documentation

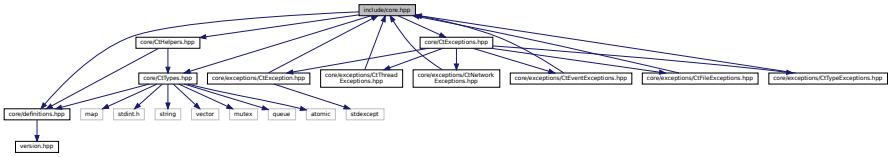
9.1 docs/mainpage.dox File Reference

9.2 docs/requirements.md File Reference

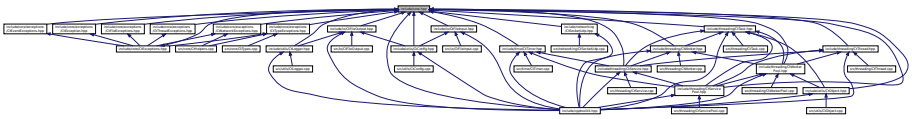
9.3 include/core.hpp File Reference

```
#include "core/definitions.hpp"
#include "core/CtTypes.hpp"
#include "core/CtHelpers.hpp"
#include "core/CtExceptions.hpp"
```

Include dependency graph for core.hpp:



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



9.3.1 Detailed Description

Date

01-02-2025

Definition in file core.hpp.

9.4 core.hpp

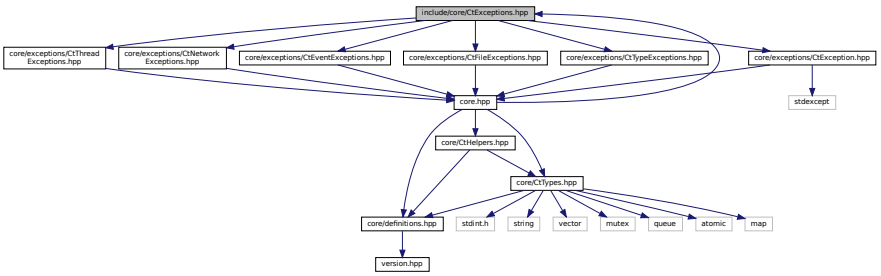
```
00001 /*
00002 MIT License
00003
00004 Copyright (c) 2024 Mouzenidis Panagiotis
00005
00006 Permission is hereby granted, free of charge, to any person obtaining a copy
00007 of this software and associated documentation files (the "Software"), to deal
00008 in the Software without restriction, including without limitation the rights
00009 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00010 copies of the Software, and to permit persons to whom the Software is
00011 furnished to do so, subject to the following conditions:
00012
00013 The above copyright notice and this permission notice shall be included in all
00014 copies or substantial portions of the Software.
00015
00016 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00017 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00018 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00019 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00020 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00021 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00022 SOFTWARE.
00023 */
00024
00032 #ifndef INCLUDE_CORE_HPP_
00033 #define INCLUDE_CORE_HPP_
00034
00035 #include "core/definitions.hpp"
00036 #include "core/CtTypes.hpp"
00037 #include "core/CtHelpers.hpp"
00038 #include "core/CtExceptions.hpp"
00039
00040 #endif //INCLUDE_CORE_HPP_
```

9.5 include/core/CtExceptions.hpp File Reference

Master header file for the exceptions in the cpptoolkit library.

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

Include dependency graph for CtExceptions.hpp:



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

9.5.1 Detailed Description

Master header file for the exceptions in the cpptoolkit library.

Date

18-01-2024

Definition in file CtExceptions.hpp.

9.6 CtExceptions.hpp

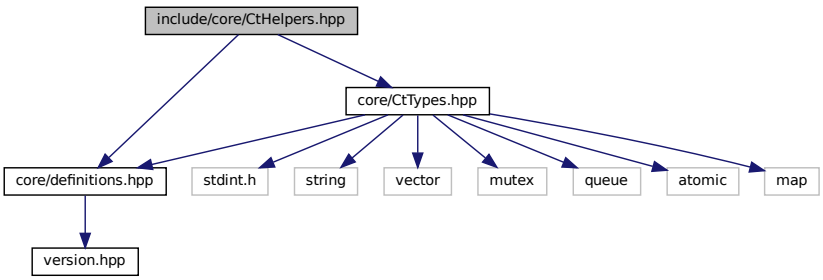
```
00001 /*
00002 MIT License
00003
00004 Copyright (c) 2024 Mouzenidis Panagiotis
00005
00006 Permission is hereby granted, free of charge, to any person obtaining a copy
00007 of this software and associated documentation files (the "Software"), to deal
00008 in the Software without restriction, including without limitation the rights
00009 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00010 copies of the Software, and to permit persons to whom the Software is
00011 furnished to do so, subject to the following conditions:
00012
00013 The above copyright notice and this permission notice shall be included in all
00014 copies or substantial portions of the Software.
00015
00016 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00017 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00018 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00019 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00020 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00021 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00022 SOFTWARE.
00023 */
00024
00032 #ifndef INCLUDE_CTEXCEPTIONS_HPP_
00033 #define INCLUDE_CTEXCEPTIONS_HPP_
00034
00035 #include "core/exceptions/CtException.hpp"
00036 #include "core/exceptions/CtThreadExceptions.hpp"
00037 #include "core/exceptions/CtNetworkExceptions.hpp"
00038 #include "core/exceptions/CtEventExceptions.hpp"
00039 #include "core/exceptions/CtFileExceptions.hpp"
00040 #include "core/exceptions/CtTypeExceptions.hpp"
00041
00042 #endif //INCLUDE_CTEXCEPTIONS_HPP_
```

9.7 include/core/CtHelpers.hpp File Reference

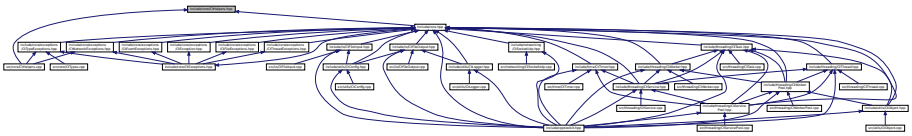
CtHelpers contains helpers for various utilities.

```
#include "core/definitions.hpp"
#include "core/CtTypes.hpp"
```

Include dependency graph for CtHelpers.hpp:



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



Namespaces

- CtStringHelpers

  This namespace contains string helper functions.
- CtSocketHelpers

  This namespace contains socket helper functions.

Macros

- #define ToCtString(x) std::to_string(x)

Functions

- EXPORTED_API void CtStringHelpers::split (const CtString &p_string, char p_delimiter, CtVector< CtString > *p_result)

  This method splits the string into substrings using the given delimiter.
- EXPORTED_API CtString CtStringHelpers::trim (const CtString &p_string)

  This method trims the string from the left and right side.
- EXPORTED_API CtDouble CtStringHelpers::StrToDouble (const CtString &p_str)

  This method converts a string to a CtDouble.
- EXPORTED_API CtFloat CtStringHelpers::StrToFloat (const CtString &p_str)

  This method converts a string to a CtFloat.
- EXPORTED_API CtUInt32 CtStringHelpers::StrToUInt (const CtString &p_str)

  This method converts a string to a CtUInt32.
- EXPORTED_API CtInt32 CtStringHelpers::StrToInt (const CtString &p_str)

  This method converts a string to a CtInt32.
- EXPORTED_API void CtSocketHelpers::setSocketTimeout (CtInt32 socketTimeout)

  Set the Socket Timeout object.
- EXPORTED_API CtVector< CtString > CtSocketHelpers::getInterfaces ()

  Get all available interfaces the device.
- EXPORTED_API CtString CtSocketHelpers::interfaceToAddress (const CtString &p_ifName)

  Get address of a specific interface.
- EXPORTED_API CtUInt32 CtSocketHelpers::getAddressAsUInt (const CtString &p_addr)

  Convert address to uin32_t.
- EXPORTED_API CtString CtSocketHelpers::getAddressAsString (CtUInt32 p_addr)

  Convert address to CtString.

Variables

- static CtInt32 CtSocketHelpers::socketTimeout = 0

9.7.1 Detailed Description

CtHelpers contains helpers for various utilities.

Date

31-01-2025

Definition in file CtHelpers.hpp.

9.7.2 Macro Definition Documentation

9.7.2.1 ToCtString

```
#define ToCtString(
              x ) std::to_string(x)
```

Convert a number to a string.

Definition at line 38 of file CtHelpers.hpp.

9.8 CtHelpers.hpp

```
00001 /*
00002 MIT License
00003
00004 Copyright (c) 2024 Mouzenidis Panagiotis
00005
00006 Permission is hereby granted, free of charge, to any person obtaining a copy
00007 of this software and associated documentation files (the "Software"), to deal
00008 in the Software without restriction, including without limitation the rights
00009 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00010 copies of the Software, and to permit persons to whom the Software is
00011 furnished to do so, subject to the following conditions:
00012
00013 The above copyright notice and this permission notice shall be included in all
00014 copies or substantial portions of the Software.
00015
00016 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00017 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00018 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00019 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00020 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00021 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00022 SOFTWARE.
00023 */
00024
00032 #ifndef INCLUDE_CTHELPERS_HPP_
00033 #define INCLUDE_CTHELPERS_HPP_
00034
00035 #include "core/definitions.hpp"
00036 #include "core/CtTypes.hpp"
00037
00038 #define ToCtString(x) std::to_string(x)
00045 namespace CtStringHelpers {
00055     EXPORTED_API void split(const CtString& p_string, char p_delimiter, CtVector<CtString> *p_result);
00056
00065     EXPORTED_API CtString trim(const CtString& p_string);
00066
00078     EXPORTED_API CtDouble StrToDouble(const CtString& p_str);
00079
00091     EXPORTED_API CtFloat StrToFloat(const CtString& p_str);
00092
```

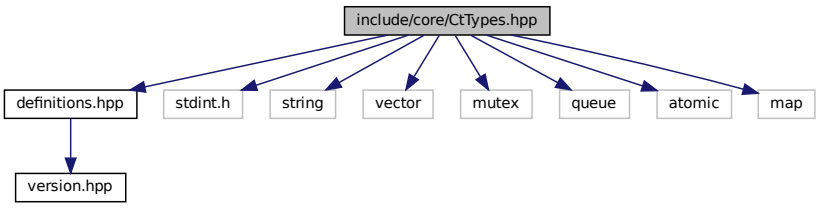
```
00104     EXPORTED_API CtUInt32 StrToUInt(const CtString& p_str);
00105
00117     EXPORTED_API CtInt32 StrToInt(const CtString& p_str);
00118 };
00119
00124 namespace CtSocketHelpers {
00125     static CtInt32 socketTimeout = 0;
00134     EXPORTED_API void setSocketTimeout(CtInt32 socketTimeout);
00135
00143     EXPORTED_API CtVector<CtString> getInterfaces();
00144
00153     EXPORTED_API CtString interfaceToAddress(const CtString& p_ifName);
00154
00163     EXPORTED_API CtUInt32 getAddressAsUInt(const CtString& p_addr);
00164
00173     EXPORTED_API CtString getAddressAsString(CtUInt32 p_addr);
00174 };
00175
00176 #endif //INCLUDE_CTHELPERS_HPP_
```

9.9 include/core/CtTypes.hpp File Reference

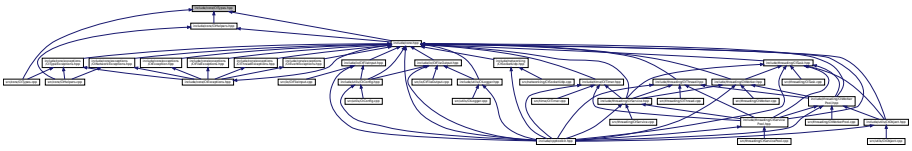
Header file for basic types and classes.

```
#include "definitions.hpp"
#include <stdint.h>
#include <string>
#include <vector>
#include <mutex>
#include <queue>
#include <atomic>
#include <map>
```

Include dependency graph for CtTypes.hpp:



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



Classes

- struct _CtNetAddress

  Struct describing a network address.
- class CtRawData

  Struct describing raw data buffer.

Macros

- #define CtUInt8 uint8_t

  Typedefs for basic types.
- #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 CtBool uint8_t
- #define CtFloat float
- #define CtDouble double
- #define CtChar char
- #define CtString std::string
- #define CtVector std::vector
- #define CtMutex std::mutex
- #define CtQueue std::queue
- #define CtAtomic std::atomic
- #define CtMap std::map
- #define CtMultiMap std::multimap
- #define CT_BUFFER_SIZE 2048u

  Default buffer size.
- #define CT_TRUE 1u
- #define CT_FALSE 0u

Typedefs

- typedef struct _CtNetAddress CtNetAddress

  Struct describing a network address.

9.9.1 Detailed Description

Header file for basic types and classes.

Date

21-01-2024

Definition in file CtTypes.hpp.

9.9.2 Macro Definition Documentation

9.9.2.1 CT_BUFFER_SIZE

```
#define CT_BUFFER_SIZE 2048u
```

Default buffer size.

Definition at line 77 of file CtTypes.hpp.

9.9.2.2 CT_FALSE

```
#define CT_FALSE 0u
```

Definition at line 80 of file CtTypes.hpp.

9.9.2.3 CT_TRUE

```
#define CT_TRUE 1u
```

Definition at line 79 of file CtTypes.hpp.

9.9.2.4 CtAtomic

```
#define CtAtomic std::atomic
```

Definition at line 69 of file CtTypes.hpp.

9.9.2.5 CtBool

```
#define CtBool uint8_t
```

Definition at line 59 of file CtTypes.hpp.

9.9.2.6 CtChar

```
#define CtChar char
```

Definition at line 64 of file CtTypes.hpp.

9.9.2.7 CtDouble

```
#define CtDouble double
```

Definition at line 62 of file CtTypes.hpp.

9.9.2.8 CtFloat

```
#define CtFloat float
```

Definition at line 61 of file CtTypes.hpp.

9.9.2.9 CtInt16

```
#define CtInt16 int16_t
```

Definition at line 55 of file CtTypes.hpp.

9.9.2.10 CtInt32

```
#define CtInt32 int32_t
```

Definition at line 56 of file CtTypes.hpp.

9.9.2.11 CtInt64

```
#define CtInt64 int64_t
```

Definition at line 57 of file CtTypes.hpp.

9.9.2.12 CtInt8

```
#define CtInt8 int8_t
```

Definition at line 54 of file CtTypes.hpp.

9.9.2.13 CtMap

```
#define CtMap std::map
```

Definition at line 70 of file CtTypes.hpp.

9.9.2.14 CtMultiMap

```
#define CtMultiMap std::multimap
```

Definition at line 71 of file CtTypes.hpp.

9.9.2.15 CtMutex

```
#define CtMutex std::mutex
```

Definition at line 67 of file CtTypes.hpp.

9.9.2.16 CtQueue

```
#define CtQueue std::queue
```

Definition at line 68 of file CtTypes.hpp.

9.9.2.17 CtString

```
#define CtString std::string
```

Definition at line 65 of file CtTypes.hpp.

9.9.2.18 CtUInt16

```
#define CtUInt16 uint16_t
```

Definition at line 50 of file CtTypes.hpp.

9.9.2.19 CtUInt32

```
#define CtUInt32 uint32_t
```

Definition at line 51 of file CtTypes.hpp.

9.9.2.20 CtUInt64

```
#define CtUInt64 uint64_t
```

Definition at line 52 of file CtTypes.hpp.

9.9.2.21 CtUInt8

```
#define CtUInt8 uint8_t
```

Typedefs for basic types.

Definition at line 49 of file CtTypes.hpp.

9.9.2.22 CtVector

```
#define CtVector std::vector
```

Definition at line 66 of file CtTypes.hpp.

9.9.3 Typedef Documentation

9.9.3.1 CtNetAddress

```
typedef struct _CtNetAddress CtNetAddress
```

Struct describing a network address.

FR-001-003-001

The network address is described by the IP address and the port number.

9.10 CtTypes.hpp

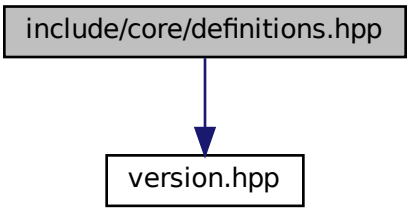
```
00001 /*
00002 MIT License
00003
00004 Copyright (c) 2024 Mouzenidis Panagiotis
00005
00006 Permission is hereby granted, free of charge, to any person obtaining a copy
00007 of this software and associated documentation files (the "Software"), to deal
00008 in the Software without restriction, including without limitation the rights
00009 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00010 copies of the Software, and to permit persons to whom the Software is
00011 furnished to do so, subject to the following conditions:
00012
00013 The above copyright notice and this permission notice shall be included in all
00014 copies or substantial portions of the Software.
00015
00016 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00017 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00018 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00019 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00020 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00021 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00022 SOFTWARE.
00023 */
00024
00032 #ifndef INCLUDE_CTTYPES_HPP_
00033 #define INCLUDE_CTTYPES_HPP_
00034
00035 #include "definitions.hpp"
00036
00037 #include <stdint.h>
00038 #include <string>
00039 #include <vector>
00040 #include <mutex>
00041 #include <queue>
00042 #include <atomic>
00043 #include <map>
00044
00049 #define CtUInt8         uint8_t
00050 #define CtUInt16        uint16_t
00051 #define CtUInt32        uint32_t
00052 #define CtUInt64        uint64_t
00053
00054 #define CtInt8          int8_t
00055 #define CtInt16         int16_t
00056 #define CtInt32         int32_t
00057 #define CtInt64         int64_t
00058
00059 #define CtBool          uint8_t
00060
00061 #define CtFloat         float
00062 #define CtDouble        double
00063
00064 #define CtChar          char
00065 #define CtString        std::string
00066 #define CtVector        std::vector
00067 #define CtMutex         std::mutex
00068 #define CtQueue         std::queue
00069 #define CtAtomic        std::atomic
00070 #define CtMap           std::map
00071 #define CtMultiMap      std::multimap
00072
00077 #define CT_BUFFER_SIZE  2048u
00078
00079 #define CT_TRUE         1u
00080 #define CT_FALSE        0u
00081
00091 typedef struct _CtNetAddress {
00092     CtString addr;
00093     CtUInt16 port;
00094 } CtNetAddress;
00095
00120 class CtRawData {
00121 public:
00131     EXPORTED_API explicit CtRawData(CtUInt32 p_size = CT_BUFFER_SIZE);
00132
00142     EXPORTED_API CtRawData(CtRawData& p_data);
00143
00150     EXPORTED_API virtual ~CtRawData();
00151
00162     EXPORTED_API void setNextByte(CtUInt8 p_data);
00163
00175     EXPORTED_API void setNextBytes(CtUInt8* p_data, CtUInt32 p_size);
00176
00187     EXPORTED_API CtUInt8* getNLastBytes(CtUInt32 p_num);
00188
```

```
00199     EXPORTED_API void removeNLastBytes(CtUInt32 p_num);
00200
00209     EXPORTED_API CtUInt32 size();
00210
00219     EXPORTED_API CtUInt32 maxSize();
00220
00228     EXPORTED_API CtUInt8* get();
00229
00242     EXPORTED_API void clone(const CtUInt8* p_data, CtUInt32 p_size);
00243
00244
00256     EXPORTED_API void clone(CtRawData& p_data);
00257
00266     EXPORTED_API void reset();
00267
00278     EXPORTED_API CtRawData& operator=(CtRawData& other);
00279
00280 private:
00281     CtUInt8* m_data;
00282     CtUInt32 m_size;
00283     const CtUInt32 m_maxSize;
00284 };
00285
00286 #endif //INCLUDE_CTTYPES_HPP_
```

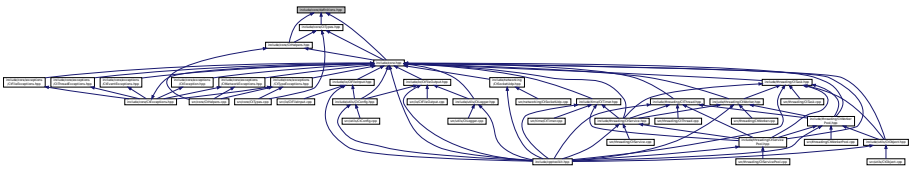
9.11 include/core/definitions.hpp File Reference

Header file for generic definitions used in teh project.

`#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")))

  EXPORTED_API macro for exporting functions in shared libraries.

9.11.1 Detailed Description

Header file for generic definitions used in teh project.

Date

18-01-2024

Definition in file definitions.hpp.

9.11.2 Macro Definition Documentation

9.11.2.1 EXPORTED_API

```
#define EXPORTED_API __attribute__((visibility("default")))
```

EXPORTED_API macro for exporting functions in shared libraries.

Definition at line 44 of file definitions.hpp.

9.12 definitions.hpp

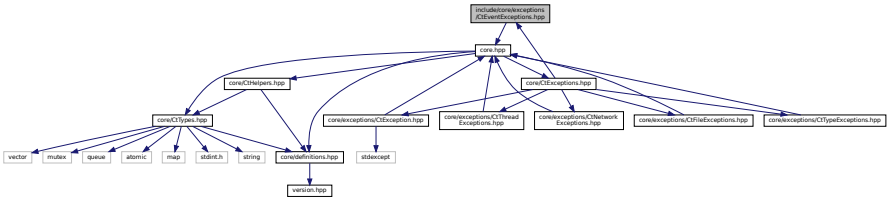
```
00001 /*
00002 MIT License
00003
00004 Copyright (c) 2024 Mouzenidis Panagiotis
00005
00006 Permission is hereby granted, free of charge, to any person obtaining a copy
00007 of this software and associated documentation files (the "Software"), to deal
00008 in the Software without restriction, including without limitation the rights
00009 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00010 copies of the Software, and to permit persons to whom the Software is
00011 furnished to do so, subject to the following conditions:
00012
00013 The above copyright notice and this permission notice shall be included in all
00014 copies or substantial portions of the Software.
00015
00016 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00017 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00018 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00019 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00020 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00021 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00022 SOFTWARE.
00023 */
00024
00032 #ifndef INCLUDE_DEFINITIONS_HPP_
00033 #define INCLUDE_DEFINITIONS_HPP_
00034
00035 #include "version.hpp"
00036
00041 #ifdef _WIN32
00042     #define EXPORTED_API __declspec(dllexport)
00043 #else
00044     #define EXPORTED_API __attribute__((visibility("default")))
00045 #endif
00046
00047 #endif //INCLUDE_DEFINITIONS_HPP_
```

9.13 include/core/exceptions/CtEventExceptions.hpp File Reference

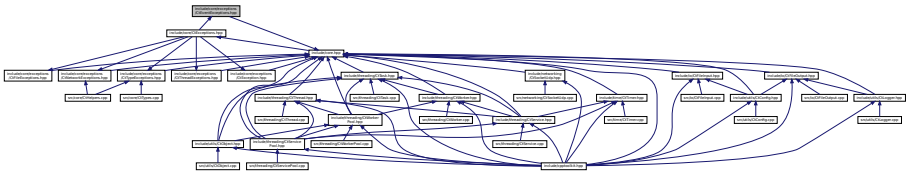
CtEventExceptions header file.

```
#include "core.hpp"
```

Include dependency graph for CtEventExceptions.hpp:



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



Classes

- class CtEventNotExistsError

  This exception is thrown when an event does not exist in the event manager.
- class CtEventAlreadyExistsError

  This exception is thrown when an event already exists in the event manager.

9.13.1 Detailed Description

CtEventExceptions header file.

Date

02-02-2024

Definition in file CtEventExceptions.hpp.

9.14 CtEventExceptions.hpp

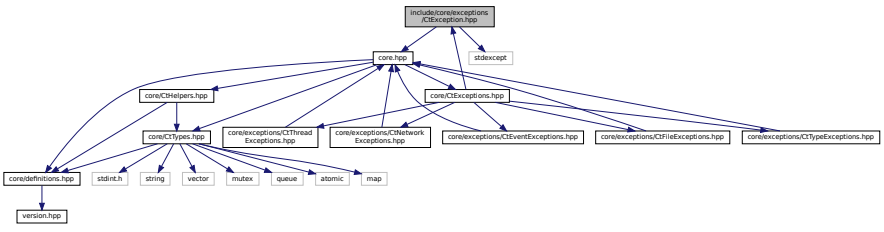
```
00001 /*
00002 MIT License
00003
00004 Copyright (c) 2024 Mouzenidis Panagiotis
00005
00006 Permission is hereby granted, free of charge, to any person obtaining a copy
00007 of this software and associated documentation files (the "Software"), to deal
00008 in the Software without restriction, including without limitation the rights
00009 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00010 copies of the Software, and to permit persons to whom the Software is
00011 furnished to do so, subject to the following conditions:
00012
00013 The above copyright notice and this permission notice shall be included in all
00014 copies or substantial portions of the Software.
00015
00016 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00017 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00018 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00019 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00020 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00021 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00022 SOFTWARE.
00023 */
00024
00032 #ifndef INCLUDE_CTEVENTEXCEPTIONS_HPP_
00033 #define INCLUDE_CTEVENTEXCEPTIONS_HPP_
00034
00035 #include "core.hpp"
00036
00044 class CtEventNotExistsError : public CtException {
00045 public:
00046     explicit CtEventNotExistsError(const CtString& msg): CtException(msg) {};
00047 };
00048
00056 class CtEventAlreadyExistsError : public CtException {
00057 public:
00058     explicit CtEventAlreadyExistsError(const CtString& msg): CtException(msg) {};
00059 };
00060
00061 #endif //INCLUDE_CTEVENTEXCEPTIONS_HPP_
```

9.15 include/core/exceptions/CtException.hpp File Reference

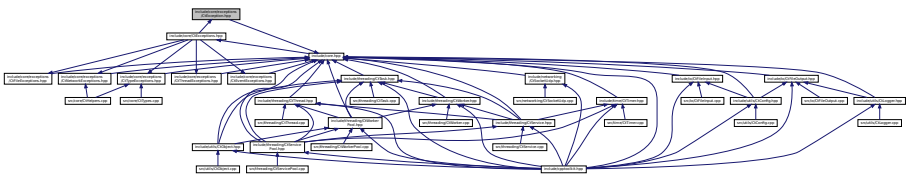
CtException header file.

```
#include "core.hpp"
#include <stdexcept>
```

Include dependency graph for CtException.hpp:



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



Classes

- class CtException

  An exception class for the cpptoolkit library.

9.15.1 Detailed Description

CtException header file.

Date

18-01-2024

Definition in file CtException.hpp.

9.16 CtException.hpp

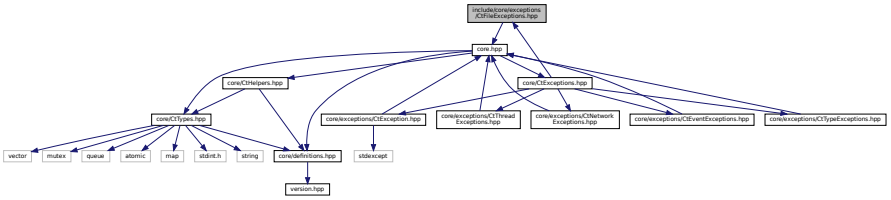
```
00001 /*
00002 MIT License
00003
00004 Copyright (c) 2024 Mouzenidis Panagiotis
00005
00006 Permission is hereby granted, free of charge, to any person obtaining a copy
00007 of this software and associated documentation files (the "Software"), to deal
00008 in the Software without restriction, including without limitation the rights
00009 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00010 copies of the Software, and to permit persons to whom the Software is
00011 furnished to do so, subject to the following conditions:
00012
00013 The above copyright notice and this permission notice shall be included in all
00014 copies or substantial portions of the Software.
00015
00016 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00017 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00018 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00019 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00020 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00021 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00022 SOFTWARE.
00023 */
00024
00032 #ifndef INCLUDE_CTEXCEPTION_HPP_
00033 #define INCLUDE_CTEXCEPTION_HPP_
00034
00035 #include "core.hpp"
00036
00037 #include <stdexcept>
00038
00047 class CtException : public std::exception {
00048 protected:
00056     explicit CtException(const CtString& msg) : m_msg(msg) {};
00057
00058 public:
00066     const char* what() const noexcept override {
00067         return m_msg.c_str();
00068     };
00069
00070 private:
00071     CtString m_msg;
00072 };
00073
00074 #endif //INCLUDE_CTEXCEPTION_HPP_
```

9.17 include/core/exceptions/CtFileExceptions.hpp File Reference

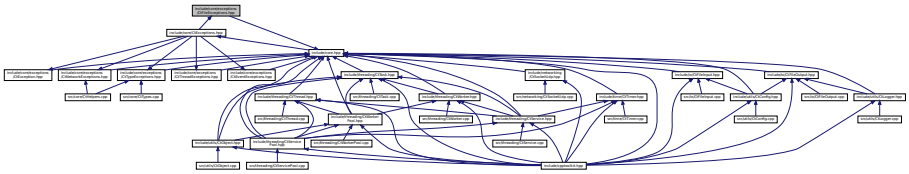
CtFileExceptions header file.

```
#include "core.hpp"
```

Include dependency graph for CtFileExceptions.hpp:



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



Classes

- class CtFileReadError

  This exception is thrown when a file cannot be read.
- class CtFileWriteError

  This exception is thrown when a file cannot be written.
- class CtFileParseError

  This exception is thrown when a file cannot be parsed.

9.17.1 Detailed Description

CtFileExceptions header file.

Date

10-03-2024

Definition in file CtFileExceptions.hpp.

9.18 CtFileExceptions.hpp

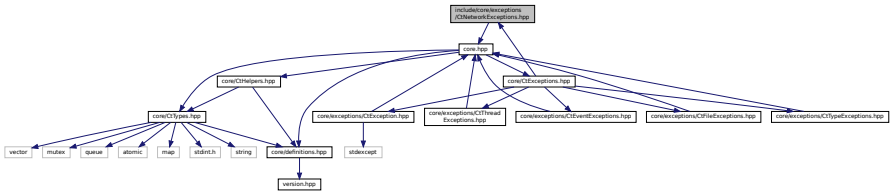
```
00001 /*
00002 MIT License
00003
00004 Copyright (c) 2024 Mouzenidis Panagiotis
00005
00006 Permission is hereby granted, free of charge, to any person obtaining a copy
00007 of this software and associated documentation files (the "Software"), to deal
00008 in the Software without restriction, including without limitation the rights
00009 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00010 copies of the Software, and to permit persons to whom the Software is
00011 furnished to do so, subject to the following conditions:
00012
00013 The above copyright notice and this permission notice shall be included in all
00014 copies or substantial portions of the Software.
00015
00016 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00017 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00018 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00019 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00020 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00021 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00022 SOFTWARE.
00023 */
00024
00032 #ifndef INCLUDE_CTFILEEXCEPTIONS_HPP_
00033 #define INCLUDE_CTFILEEXCEPTIONS_HPP_
00034
00035 #include "core.hpp"
00036
00044 class CtFileReadError : public CtException {
00045 public:
00046     explicit CtFileReadError(const CtString& msg): CtException(msg) {};
00047 };
00048
00056 class CtFileWriteError : public CtException {
00057 public:
00058     explicit CtFileWriteError(const CtString& msg): CtException(msg) {};
00059 };
00060
00068 class CtFileParseError : public CtException {
00069 public:
00070     explicit CtFileParseError(const CtString& msg): CtException(msg) {};
00071 };
00072
00073 #endif //INCLUDE_CTFILEEXCEPTIONS_HPP_
```

9.19 include/core/exceptions/CtNetworkExceptions.hpp File Reference

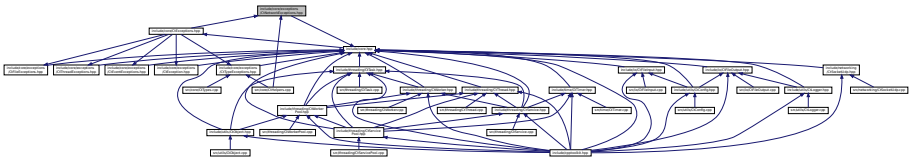
CtNetworkExceptions header file.

```
#include "core.hpp"
```

Include dependency graph for CtNetworkExceptions.hpp:



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



Classes

- class CtSocketError

  This exception is thrown when a socket error occurs.
- class CtSocketBindError

  This exception is thrown when a socket bind error occurs.
- class CtSocketPollError

  This exception is thrown when a socket listen error occurs.
- class CtSocketReadError

  This exception is thrown when a socket accept error occurs.
- class CtSocketWriteError

  This exception is thrown when a socket connect error occurs.

9.19.1 Detailed Description

CtNetworkExceptions header file.

Date

18-01-2024

Definition in file CtNetworkExceptions.hpp.

9.20 CtNetworkExceptions.hpp

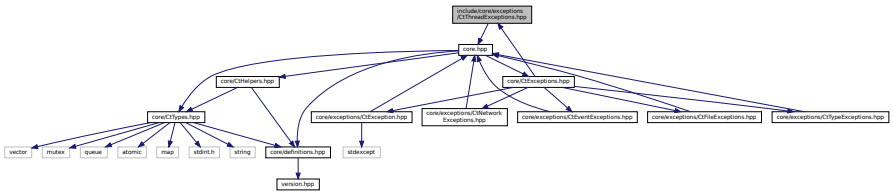
```
00001 /*
00002 MIT License
00003
00004 Copyright (c) 2024 Mouzenidis Panagiotis
00005
00006 Permission is hereby granted, free of charge, to any person obtaining a copy
00007 of this software and associated documentation files (the "Software"), to deal
00008 in the Software without restriction, including without limitation the rights
00009 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00010 copies of the Software, and to permit persons to whom the Software is
00011 furnished to do so, subject to the following conditions:
00012
00013 The above copyright notice and this permission notice shall be included in all
00014 copies or substantial portions of the Software.
00015
00016 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00017 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00018 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00019 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00020 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00021 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00022 SOFTWARE.
00023 */
00024
00032 #ifndef INCLUDE_CTNETWORKEXCEPTIONS_HPP_
00033 #define INCLUDE_CTNETWORKEXCEPTIONS_HPP_
00034
00035 #include "core.hpp"
00036
00044 class CtSocketError : public CtException {
00045 public:
00046     explicit CtSocketError(const CtString& msg): CtException(msg) {};
00047 };
00048
00056 class CtSocketBindError : public CtException {
00057 public:
00058     explicit CtSocketBindError(const CtString& msg): CtException(msg) {};
00059 };
00060
00068 class CtSocketPollError : public CtException {
00069 public:
```

```
00070     explicit CtSocketPollError(const CtString& msg): CtException(msg) {};
00071 };
00072
00080 class CtSocketReadError : public CtException {
00081 public:
00082     explicit CtSocketReadError(const CtString& msg): CtException(msg) {};
00083 };
00084
00092 class CtSocketWriteError : public CtException {
00093 public:
00094     explicit CtSocketWriteError(const CtString& msg): CtException(msg) {};
00095 };
00096
00097 #endif //INCLUDE_CTNETWORKEXCEPTIONS_HPP_
```

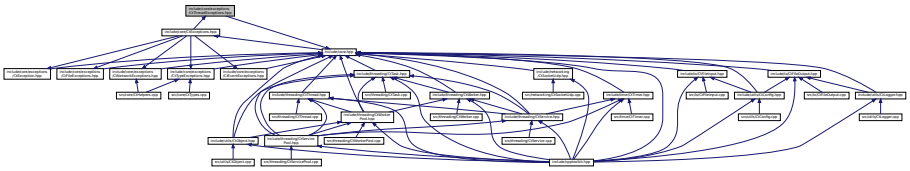
9.21 include/core/exceptions/CtThreadExceptions.hpp File Reference

CtThreadExceptions header file.

`#include "core.hpp"`
Include dependency graph for CtThreadExceptions.hpp:



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



Classes

- class CtThreadError

  This exception is thrown when a thread error occurs.
- class CtServiceError

  This exception is thrown when a service pool error occurs.
- class CtWorkerError

  This exception is thrown when a worker error occurs.

9.21.1 Detailed Description

CtThreadExceptions header file.

Date

18-01-2024

Definition in file CtThreadExceptions.hpp.

9.22 CtThreadExceptions.hpp

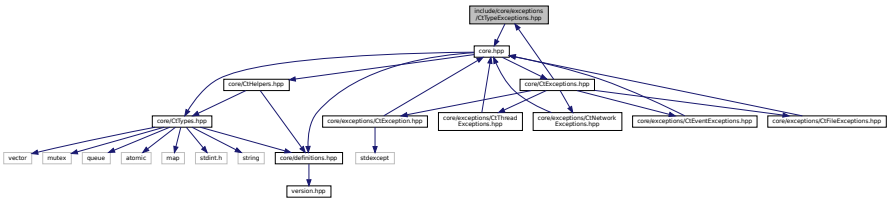
```
00001 /*
00002 MIT License
00003
00004 Copyright (c) 2024 Mouzenidis Panagiotis
00005
00006 Permission is hereby granted, free of charge, to any person obtaining a copy
00007 of this software and associated documentation files (the "Software"), to deal
00008 in the Software without restriction, including without limitation the rights
00009 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00010 copies of the Software, and to permit persons to whom the Software is
00011 furnished to do so, subject to the following conditions:
00012
00013 The above copyright notice and this permission notice shall be included in all
00014 copies or substantial portions of the Software.
00015
00016 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00017 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00018 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00019 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00020 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00021 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00022 SOFTWARE.
00023 */
00024
00032 #ifndef INCLUDE_CTTHREADEXCEPTIONS_HPP_
00033 #define INCLUDE_CTTHREADEXCEPTIONS_HPP_
00034
00035 #include "core.hpp"
00036
00044 class CtThreadError : public CtException {
00045 public:
00046     explicit CtThreadError(const CtString& msg): CtException(msg) {};
00047 };
00048
00056 class CtServiceError : public CtException {
00057 public:
00058     explicit CtServiceError(const CtString& msg): CtException(msg) {};
00059 };
00060
00068 class CtWorkerError : public CtException {
00069 public:
00070     explicit CtWorkerError(const CtString& msg): CtException(msg) {};
00071 };
00072
00073 #endif //INCLUDE_CTTHREADEXCEPTIONS_HPP_
```

9.23 include/core/exceptions/CtTypeExceptions.hpp File Reference

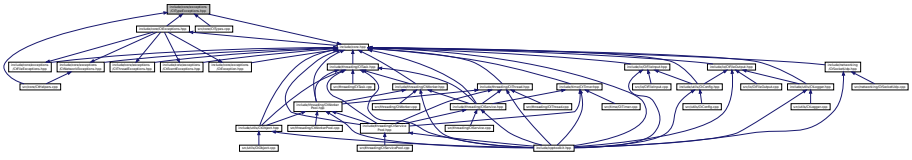
CtTypeExceptions header file.

`#include "core.hpp"`

Include dependency graph for CtTypeExceptions.hpp:



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



Classes

- class CtTypeParseError

  This exception is thrown when a type cannot be parsed.
- class CtKeyNotFoundError

  This exception is thrown when a key is not found in a container.
- class CtOutOfRangeError

  This exception is thrown when an index is out of bounds.

9.23.1 Detailed Description

CtTypeExceptions header file.

Date

10-03-2024

Definition in file CtTypeExceptions.hpp.

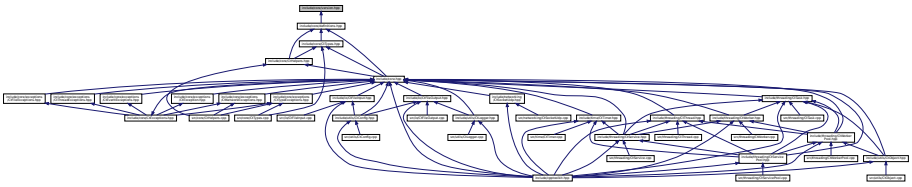
9.24 CtTypeExceptions.hpp

```
00001 /*
00002 MIT License
00003
00004 Copyright (c) 2024 Mouzenidis Panagiotis
00005
00006 Permission is hereby granted, free of charge, to any person obtaining a copy
00007 of this software and associated documentation files (the "Software"), to deal
00008 in the Software without restriction, including without limitation the rights
00009 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00010 copies of the Software, and to permit persons to whom the Software is
00011 furnished to do so, subject to the following conditions:
00012
00013 The above copyright notice and this permission notice shall be included in all
00014 copies or substantial portions of the Software.
00015
00016 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00017 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00018 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00019 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00020 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00021 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00022 SOFTWARE.
00023 */
00024
00032 #ifndef INCLUDE_CTTYPEEXCEPTIONS_HPP_
00033 #define INCLUDE_CTTYPEEXCEPTIONS_HPP_
00034
00035 #include "core.hpp"
00036
00044 class CtTypeParseError : public CtException {
00045 public:
00046     explicit CtTypeParseError(const CtString& msg): CtException(msg) {};
00047 };
00048
00056 class CtKeyNotFoundError : public CtException {
00057 public:
00058     explicit CtKeyNotFoundError(const CtString& msg): CtException(msg) {};
00059 };
00060
00068 class CtOutOfRangeError : public CtException {
00069 public:
00070     explicit CtOutOfRangeError(const CtString& msg): CtException(msg) {};
00071 };
00072
00073 #endif //INCLUDE_CTTYPEEXCEPTIONS_HPP_
```

9.25 include/core/version.hpp File Reference

Version information for the project.

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



Macros

- #define CPPTOOLKIT_VERSION_MAJOR 0

  Version information for the project.
- #define CPPTOOLKIT_VERSION_MINOR 1
- #define CPPTOOLKIT_VERSION_PATCH 0
- #define CPPTOOLKIT_VERSION (CPPTOOLKIT_VERSION_MAJOR ## "." ## CPPTOOLKIT_VERSION_MINOR ## "." ## CPPTOOLKIT_VERSION_PATCH)

9.25.1 Detailed Description

Version information for the project.

Date

18-01-2024

Definition in file version.hpp.

9.25.2 Macro Definition Documentation

9.25.2.1 CPPTOOLKIT_VERSION

```
#define CPPTOOLKIT_VERSION (CPPTOOLKIT_VERSION_MAJOR ## "." ## CPPTOOLKIT_VERSION_MINOR ## "."
## CPPTOOLKIT_VERSION_PATCH)
```

Definition at line 50 of file version.hpp.

9.25.2.2 CPPTOOLKIT_VERSION_MAJOR

```
#define CPPTOOLKIT_VERSION_MAJOR 0
```

Version information for the project.

The version information is defined by three macros:

- CPPTOOLKIT_VERSION_MAJOR
- CPPTOOLKIT_VERSION_MINOR
- CPPTOOLKIT_VERSION_PATCH These macros has to be modified after a new release.

Definition at line 46 of file version.hpp.

9.25.2.3 CPPTOOLKIT_VERSION_MINOR

```
#define CPPTOOLKIT_VERSION_MINOR 1
```

Definition at line 47 of file version.hpp.

9.25.2.4 CPPTOOLKIT_VERSION_PATCH

```
#define CPPTOOLKIT_VERSION_PATCH 0
```

Definition at line 48 of file version.hpp.

9.26 version.hpp

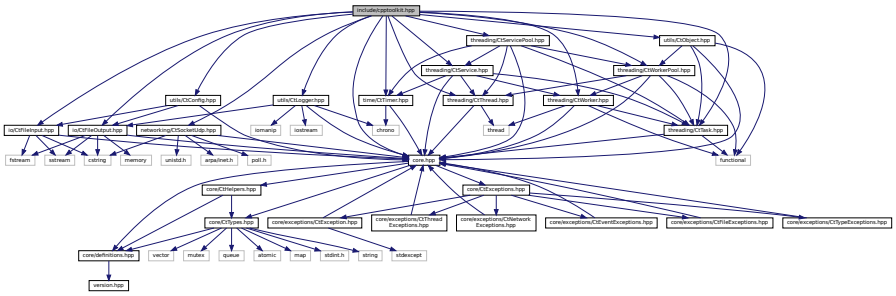
```
00001 /*
00002 MIT License
00003
00004 Copyright (c) 2024 Mouzenidis Panagiotis
00005
00006 Permission is hereby granted, free of charge, to any person obtaining a copy
00007 of this software and associated documentation files (the "Software"), to deal
00008 in the Software without restriction, including without limitation the rights
00009 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00010 copies of the Software, and to permit persons to whom the Software is
00011 furnished to do so, subject to the following conditions:
00012
00013 The above copyright notice and this permission notice shall be included in all
00014 copies or substantial portions of the Software.
00015
00016 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00017 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00018 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00019 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00020 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00021 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00022 SOFTWARE.
00023 */
00024
00032 #ifndef INCLUDE_VERSION_HPP_
00033 #define INCLUDE_VERSION_HPP_
00034
00046 #define CPPTOOLKIT_VERSION_MAJOR 0
00047 #define CPPTOOLKIT_VERSION_MINOR 1
00048 #define CPPTOOLKIT_VERSION_PATCH 0
00049
00050 #define CPPTOOLKIT_VERSION       (CPPTOOLKIT_VERSION_MAJOR ## "." ## CPPTOOLKIT_VERSION_MINOR ## "."
       ## CPPTOOLKIT_VERSION_PATCH)
00051
00052 #endif //INCLUDE_VERSION_HPP_
```

9.27 include/cpptoolkit.hpp File Reference

Master header file for the C++ Toolkit library.

```
#include "core.hpp"
#include "io/CtFileInput.hpp"
#include "io/CtFileOutput.hpp"
#include "networking/CtSocketUdp.hpp"
#include "threading/CtTask.hpp"
#include "threading/CtThread.hpp"
#include "threading/CtWorker.hpp"
#include "threading/CtWorkerPool.hpp"
#include "threading/CtService.hpp"
#include "threading/CtServicePool.hpp"
#include "time/CtTimer.hpp"
#include "utils/CtConfig.hpp"
#include "utils/CtLogger.hpp"
#include "utils/CtObject.hpp"
```

Include dependency graph for cpptoolkit.hpp:



9.27.1 Detailed Description

Master header file for the C++ Toolkit library.

Date

10-01-2025

Definition in file cpptoolkit.hpp.

9.28 cpptoolkit.hpp

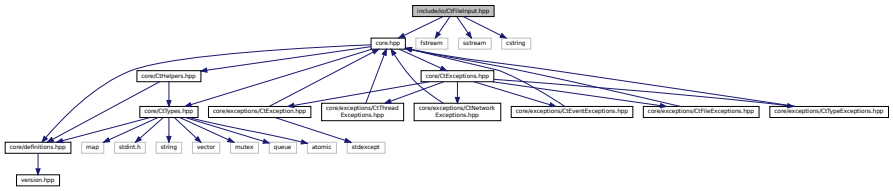
```
00001 /*
00002 MIT License
00003
00004 Copyright (c) 2024 Mouzenidis Panagiotis
00005
00006 Permission is hereby granted, free of charge, to any person obtaining a copy
00007 of this software and associated documentation files (the "Software"), to deal
00008 in the Software without restriction, including without limitation the rights
00009 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00010 copies of the Software, and to permit persons to whom the Software is
00011 furnished to do so, subject to the following conditions:
00012
00013 The above copyright notice and this permission notice shall be included in all
```

```
00014 copies or substantial portions of the Software.
00015
00016 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00017 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00018 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00019 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00020 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00021 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00022 SOFTWARE.
00023 */
00024
00032 #ifndef INCLUDE_CPPTOOLKIT_HPP_
00033 #define INCLUDE_CPPTOOLKIT_HPP_
00034
00039 #include "core.hpp"
00040
00045 #include "io/CtFileInput.hpp"
00046 #include "io/CtFileOutput.hpp"
00047
00052 #include "networking/CtSocketUdp.hpp"
00053
00058 #include "threading/CtTask.hpp"
00059 #include "threading/CtThread.hpp"
00060 #include "threading/CtWorker.hpp"
00061 #include "threading/CtWorkerPool.hpp"
00062 #include "threading/CtService.hpp"
00063 #include "threading/CtServicePool.hpp"
00064
00069 #include "time/CtTimer.hpp"
00070
00075 #include "utils/CtConfig.hpp"
00076 #include "utils/CtLogger.hpp"
00077 #include "utils/CtObject.hpp"
00078
00079 #endif //INCLUDE_CPPTOOLKIT_HPP_
```

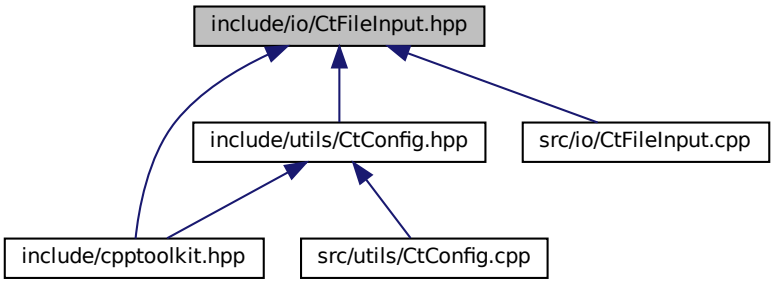
9.29 include/io/CtFileInput.hpp File Reference

```
#include "core.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

  CtFileInput class for reading data from file.

9.29.1 Detailed Description

Date

08-03-2024

Definition in file CtFileInput.hpp.

9.30 CtFileInput.hpp

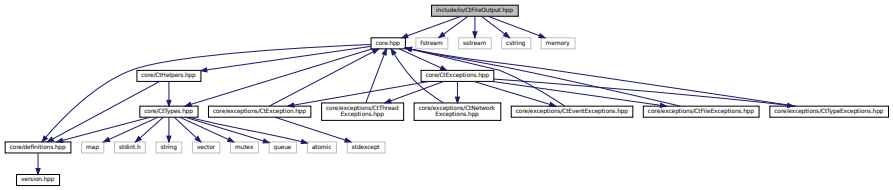
```
00001 /*
00002 MIT License
00003
00004 Copyright (c) 2024 Mouzenidis Panagiotis
00005
00006 Permission is hereby granted, free of charge, to any person obtaining a copy
00007 of this software and associated documentation files (the "Software"), to deal
00008 in the Software without restriction, including without limitation the rights
00009 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00010 copies of the Software, and to permit persons to whom the Software is
00011 furnished to do so, subject to the following conditions:
00012
00013 The above copyright notice and this permission notice shall be included in all
00014 copies or substantial portions of the Software.
00015
00016 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00017 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00018 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00019 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00020 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00021 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00022 SOFTWARE.
00023 */
00024
00032 #ifndef INCLUDE_CTFILEINPUT_HPP_
00033 #define INCLUDE_CTFILEINPUT_HPP_
00034
00035 #include "core.hpp"
00036
00037 #include <fstream>
00038 #include <sstream>
```

```
00039 #include <cstring>
00040
00057 class CtFileInput {
00058 public:
00068     EXPORTED_API explicit CtFileInput(const CtString& p_fileName);
00069
00077     EXPORTED_API ~CtFileInput();
00078
00087     EXPORTED_API void setDelimiter(const CtChar* p_delim, CtUInt8 p_delim_size);
00088
00102     EXPORTED_API CtBool read(CtRawData* p_data);
00103
00104 private:
00105     std::ifstream m_file;
00106     CtChar* m_delim;
00107     CtUInt8 m_delim_size;
00108 };
00109
00110 #endif //INCLUDE_CTFILEINPUT_HPP_
```

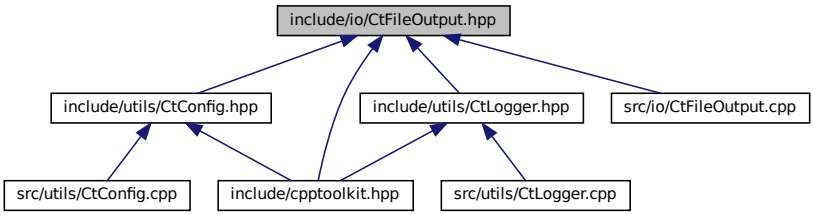
9.31 include/io/CtFileOutput.hpp File Reference

```
#include "core.hpp"
#include <fstream>
#include <sstream>
#include <cstring>
#include <memory>
```

Include dependency graph for CtFileOutput.hpp:



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



Classes

- class CtFileOutput

  CtFileOutput class for writing data to file.

9.31.1 Detailed Description

Date

09-03-2024

Definition in file CtFileOutput.hpp.

9.32 CtFileOutput.hpp

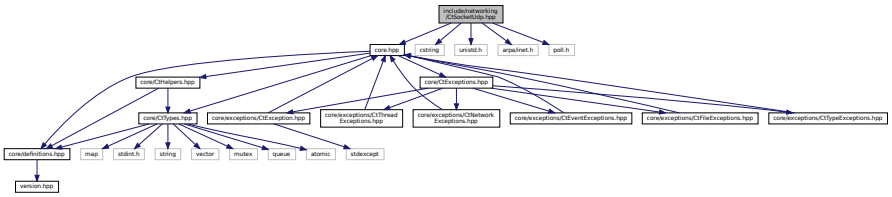
```
00001 /*
00002 MIT License
00003
00004 Copyright (c) 2024 Mouzenidis Panagiotis
00005
00006 Permission is hereby granted, free of charge, to any person obtaining a copy
00007 of this software and associated documentation files (the "Software"), to deal
00008 in the Software without restriction, including without limitation the rights
00009 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00010 copies of the Software, and to permit persons to whom the Software is
00011 furnished to do so, subject to the following conditions:
00012
00013 The above copyright notice and this permission notice shall be included in all
00014 copies or substantial portions of the Software.
00015
00016 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00017 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00018 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00019 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00020 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00021 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00022 SOFTWARE.
00023 */
00024
00032 #ifndef INCLUDE_CTFILEOUTPUT_HPP_
00033 #define INCLUDE_CTFILEOUTPUT_HPP_
00034
00035 #include "core.hpp"
00036
00037 #include <fstream>
00038 #include <sstream>
00039 #include <cstring>
00040 #include <memory>
00041
00055 class CtFileOutput {
00056 public:
00062     enum class WriteMode { Append, Truncate };
00063
00074     EXPORTED_API explicit CtFileOutput(const CtString& p_fileName, WriteMode p_mode =
      WriteMode::Append);
00075
00083     EXPORTED_API ~CtFileOutput();
00084
00093     EXPORTED_API void setDelimiter(const char* p_delim, CtUInt8 p_delim_size);
00094
00108     EXPORTED_API void write(CtRawData* p_data);
00109
00122     EXPORTED_API void writePart(CtRawData* p_data);
00123
00124 private:
00125     std::ofstream m_file;
00126     std::unique_ptr<char[]> m_delim;
00127     CtUInt8 m_delim_size;
00128 };
00129
00130
00131 #endif //INCLUDE_CTFILEOUTPUT_HPP_
```

9.33 include/networking/CtSocketUdp.hpp File Reference

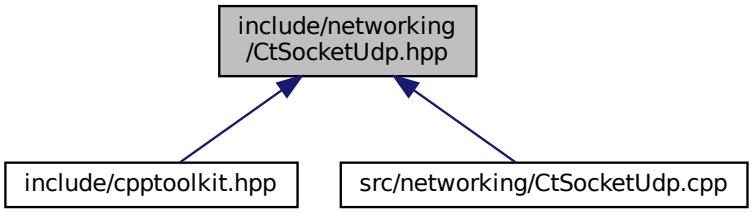
CtSocketUdp class header file.

```
#include "core.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.

9.33.1 Detailed Description

CtSocketUdp class header file.

Date

18-01-2024

Definition in file CtSocketUdp.hpp.

9.34 CtSocketUdp.hpp

```
00001 /*
00002 MIT License
00003
00004 Copyright (c) 2024 Mouzenidis Panagiotis
00005
00006 Permission is hereby granted, free of charge, to any person obtaining a copy
00007 of this software and associated documentation files (the "Software"), to deal
00008 in the Software without restriction, including without limitation the rights
00009 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00010 copies of the Software, and to permit persons to whom the Software is
00011 furnished to do so, subject to the following conditions:
00012
00013 The above copyright notice and this permission notice shall be included in all
00014 copies or substantial portions of the Software.
00015
00016 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00017 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00018 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00019 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00020 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00021 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00022 SOFTWARE.
00023 */
00024
00032 #ifndef INCLUDE_CTSOCKETUDP_HPP_
00033 #define INCLUDE_CTSOCKETUDP_HPP_
00034
00035 #include "core.hpp"
00036
00037 #include <cstring>
00038 #include <unistd.h>
00039 #include <arpa/inet.h>
00040 #include <poll.h>
00041
00082 class CtSocketUdp {
00083 public:
00091     EXPORTED_API CtSocketUdp();
00092
00099     EXPORTED_API ~CtSocketUdp();
00100
00109     EXPORTED_API void setSub(const CtString& p_interfaceName, CtUInt16 p_port);
00110
00119     EXPORTED_API void setPub(CtUInt16 p_port, const CtString& p_addr = "0.0.0.0");
00120
00129     EXPORTED_API CtBool pollRead();
00130
00139     EXPORTED_API CtBool pollWrite();
00140
00150     EXPORTED_API void send(CtUInt8* p_data, CtUInt32 p_size);
00151
00160     EXPORTED_API void send(CtRawData& p_message);
00161
00172     EXPORTED_API void receive(CtUInt8* p_data, CtUInt32 p_size, CtNetAddress* p_client = nullptr);
00173
00183     EXPORTED_API void receive(CtRawData* p_message, CtNetAddress* p_clientAddress = nullptr);
00184
00185 private:
00186     int m_addrType;
00187     int m_socket;
00188     CtUInt16 m_port;
00189     CtString m_addr;
00190     struct pollfd m_pollin_sockets[1];
00191     struct pollfd m_pollout_sockets[1];
00192     sockaddr_in m_pubAddress;
00193     sockaddr_in m_subAddress;
00194 };
00195
00196 #endif //INCLUDE_CTSOCKETUDP_HPP_
```

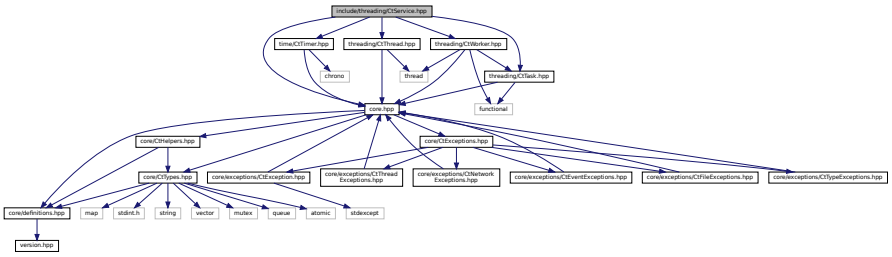
9.35 include/threading/CtService.hpp File Reference

CtService class header file.

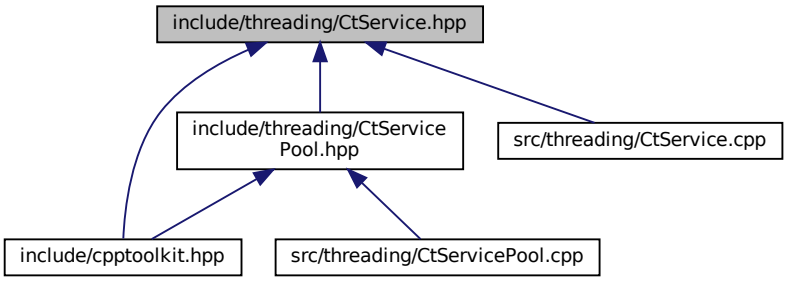
```
#include "core.hpp"
#include "threading/CtThread.hpp"
#include "threading/CtWorker.hpp"
#include "threading/CtTask.hpp"
```

```
#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.

9.35.1 Detailed Description

CtService class header file.

Date

18-01-2024

Definition in file CtService.hpp.

9.36 CtService.hpp

```
00001 /*
00002 MIT License
00003
00004 Copyright (c) 2024 Mouzenidis Panagiotis
00005
00006 Permission is hereby granted, free of charge, to any person obtaining a copy
00007 of this software and associated documentation files (the "Software"), to deal
00008 in the Software without restriction, including without limitation the rights
00009 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00010 copies of the Software, and to permit persons to whom the Software is
00011 furnished to do so, subject to the following conditions:
00012
00013 The above copyright notice and this permission notice shall be included in all
00014 copies or substantial portions of the Software.
00015
00016 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00017 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00018 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00019 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00020 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00021 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00022 SOFTWARE.
00023 */
00024
00032 #ifndef INCLUDE_CTSERVICE_HPP_
00033 #define INCLUDE_CTSERVICE_HPP_
00034
00035 #include "core.hpp"
00036
00037 #include "threading/CtThread.hpp"
00038 #include "threading/CtWorker.hpp"
00039 #include "threading/CtTask.hpp"
00040 #include "time/CtTimer.hpp"
00041
00064 class CtService : private CtThread {
00065 public:
00074     EXPORTED_API CtService(CtUInt64 nslots, const CtTask& task);
00075
00085     template <typename F, typename... FArgs>
00086     EXPORTED_API CtService(CtUInt64 nslots, const F&& func, FArgs&&... fargs);
00087
00095     EXPORTED_API ~CtService();
00096
00107     EXPORTED_API void runService();
00108
00115     EXPORTED_API void stopService();
00116
00130     EXPORTED_API float getIntervalValidity();
00131
00132 public:
00139     static CtUInt32 m_slot_time;
00140
00141 private:
00150     void loop() override;
00151
00152 private:
00153     CtWorker m_worker;
00154     CtUInt64 m_nslots;
00155     CtUInt32 m_skip_ctr;
00156     CtUInt32 m_exec_ctr;
00157 };
00158
00159 template <typename F, typename... FArgs>
00160 CtService::CtService(CtUInt64 nslots, const F&& func, FArgs&&... fargs) : m_nslots(nslots){
00161     CtTask s_task;
00162     s_task.setTaskFunc(std::bind(func, std::forward<FArgs>(fargs)...));
00163     m_worker.setTask(s_task);
00164 };
00165
00166 #endif //INCLUDE_CTSERVICE_HPP_
```

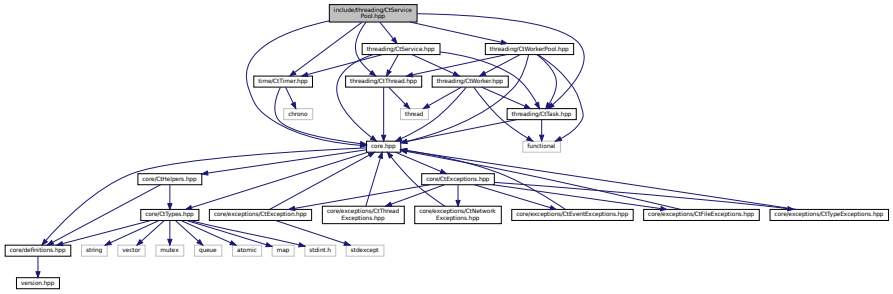
9.37 include/threading/CtServicePool.hpp File Reference

CtServicePool class header file.

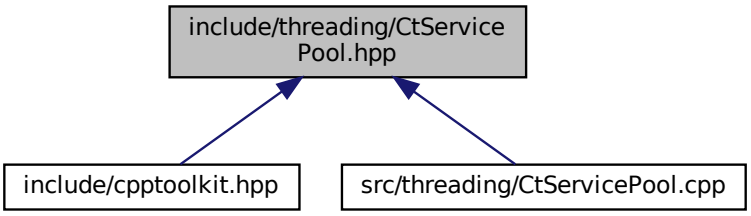
```
#include "core.hpp"
#include "threading/CtService.hpp"
```

```
#include "threading/CtWorkerPool.hpp"
#include "threading/CtThread.hpp"
#include "time/CtTimer.hpp"
#include "threading/CtTask.hpp"
```

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.
- struct CtServicePool::_CtServicePack

9.37.1 Detailed Description

CtServicePool class header file.

Date

18-01-2024

Definition in file CtServicePool.hpp.

9.38 CtServicePool.hpp

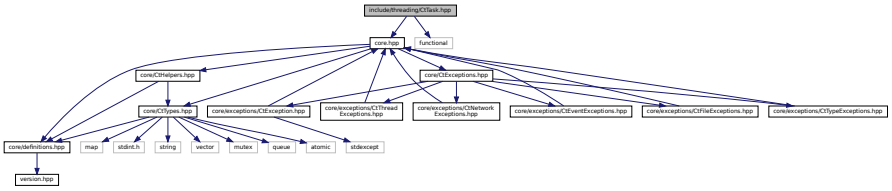
```
00001 /*
00002 MIT License
00003
00004 Copyright (c) 2024 Mouzenidis Panagiotis
00005
00006 Permission is hereby granted, free of charge, to any person obtaining a copy
00007 of this software and associated documentation files (the "Software"), to deal
00008 in the Software without restriction, including without limitation the rights
00009 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00010 copies of the Software, and to permit persons to whom the Software is
00011 furnished to do so, subject to the following conditions:
00012
00013 The above copyright notice and this permission notice shall be included in all
00014 copies or substantial portions of the Software.
00015
00016 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00017 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00018 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00019 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00020 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00021 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00022 SOFTWARE.
00023 */
00024
00032 #ifndef INCLUDE_CTSERVICEPOOL_HPP_
00033 #define INCLUDE_CTSERVICEPOOL_HPP_
00034
00035 #include "core.hpp"
00036
00037 #include "threading/CtService.hpp"
00038 #include "threading/CtWorkerPool.hpp"
00039 #include "threading/CtThread.hpp"
00040 #include "time/CtTimer.hpp"
00041 #include "threading/CtTask.hpp"
00042
00070 class CtServicePool : private CtThread {
00071 private:
00078     typedef struct _CtServicePack {
00079         CtTask task;
00080         CtString id;
00081         CtUInt32 nslots;
00082     } CtServicePack;
00083
00084 public:
00092     EXPORTED_API explicit CtServicePool(CtUInt32 nworkers);
00093
00101     EXPORTED_API ~CtServicePool();
00102
00116     EXPORTED_API void addTask(CtUInt32 nslots, const CtString& id, CtTask& task);
00117
00132     template <typename F, typename... FArgs>
00133     EXPORTED_API void addTaskFunc(CtUInt32 nslots, const CtString& id, F&& func, FArgs&&... fargs);
00134
00142     EXPORTED_API void removeTask(const CtString& id);
00143
00150     EXPORTED_API void startServices();
00151
00158     EXPORTED_API void shutdownServices();
00159
00160 private:
00167     void loop() override;
00168
00169 private:
00170     CtUInt32 m_nworkers;
00171     CtUInt32 m_slot_cnt;
00172     CtVector<CtServicePack> m_tasks;
00173     CtMutex m_mtx_control;
00174     CtWorkerPool m_worker_pool;
00175     CtTimer m_timer;
00176     CtUInt64 m_exec_time;
00177 };
00178
00179 template <typename F, typename... FArgs>
00180 void CtServicePool::addTaskFunc(CtUInt32 nslots, const CtString& id, F&& func, FArgs&&... fargs) {
00181     CtTask s_task;
00182     s_task.setTaskFunc(std::bind(func, std::forward<FArgs>(fargs)...));
00183     addTask(nslots, id, s_task);
00184 };
00185
00186 #endif //INCLUDE_CTSERVICEPOOL_HPP_
```

9.39 include/threading/CtTask.hpp File Reference

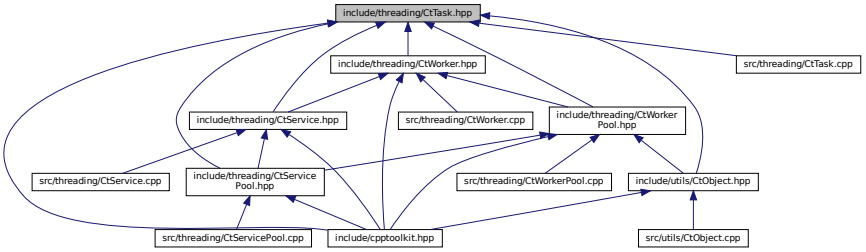
CtTask class header file.

```
#include "core.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.

9.39.1 Detailed Description

CtTask class header file.

Date

18-01-2024

Definition in file CtTask.hpp.

9.40 CtTask.hpp

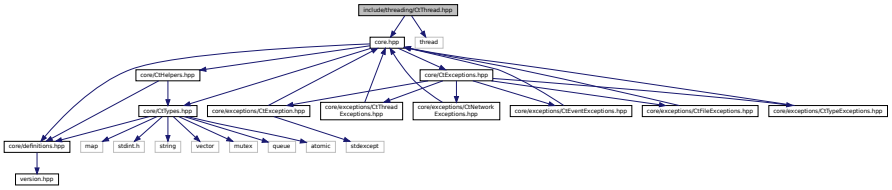
```
00001 /*
00002 MIT License
00003
00004 Copyright (c) 2024 Mouzenidis Panagiotis
00005
00006 Permission is hereby granted, free of charge, to any person obtaining a copy
00007 of this software and associated documentation files (the "Software"), to deal
00008 in the Software without restriction, including without limitation the rights
00009 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00010 copies of the Software, and to permit persons to whom the Software is
00011 furnished to do so, subject to the following conditions:
00012
00013 The above copyright notice and this permission notice shall be included in all
00014 copies or substantial portions of the Software.
00015
00016 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00017 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00018 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00019 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00020 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00021 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00022 SOFTWARE.
00023 */
00024
00032 #ifndef INCLUDE_CTTASK_HPP_
00033 #define INCLUDE_CTTASK_HPP_
00034
00035 #include "core.hpp"
00036
00037 #include <functional>
00038
00061 class CtTask {
00062 public:
00070     EXPORTED_API explicit CtTask();
00071
00082     EXPORTED_API CtTask(const CtTask& other);
00083
00089     EXPORTED_API ~CtTask();
00090
00105     template <typename F, typename... FArgs>
00106     EXPORTED_API void setTaskFunc(const F&& func, FArgs&&... fargs);
00107
00122     template <typename C, typename... CArgs>
00123     EXPORTED_API void setCallbackFunc(const C&& callback, CArgs&&... cargs);
00124
00133     EXPORTED_API std::function<void()> getTaskFunc();
00134
00143     EXPORTED_API std::function<void()> getCallbackFunc();
00144
00156     EXPORTED_API CtTask& operator=(const CtTask& other);
00157
00158 private:
00159     std::function<void()> m_task;
00160     std::function<void()> m_callback;
00161 };
00162
00163 template <typename F, typename... FArgs>
00164 void CtTask::setTaskFunc(const F&& func, FArgs&&... fargs) {
00165     m_task = std::bind(func, std::forward<FArgs>(fargs)...);
00166 };
00167
00168 template <typename C, typename... CArgs>
00169 void CtTask::setCallbackFunc(const C&& callback, CArgs&&... cargs) {
00170     m_callback = std::bind(callback, std::forward<CArgs>(cargs)...);
00171 };
00172
00173 #endif //INCLUDE_CTTASK_HPP_
```

9.41 include/threading/CtThread.hpp File Reference

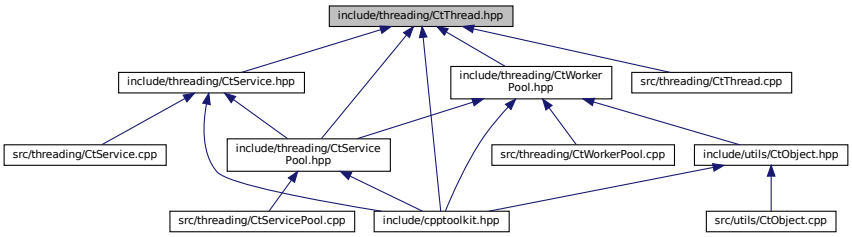
CtThread class header file.

```
#include "core.hpp"
#include <thread>
```

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.

9.41.1 Detailed Description

CtThread class header file.

Date

18-01-2024

Definition in file CtThread.hpp.

9.42 CtThread.hpp

```
00001 /*
00002 MIT License
00003
00004 Copyright (c) 2024 Mouzenidis Panagiotis
00005
00006 Permission is hereby granted, free of charge, to any person obtaining a copy
00007 of this software and associated documentation files (the "Software"), to deal
00008 in the Software without restriction, including without limitation the rights
00009 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00010 copies of the Software, and to permit persons to whom the Software is
00011 furnished to do so, subject to the following conditions:
00012
00013 The above copyright notice and this permission notice shall be included in all
00014 copies or substantial portions of the Software.
```

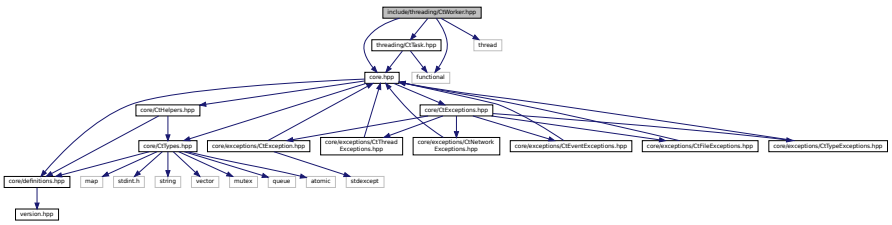
```
00015
00016 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00017 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00018 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00019 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00020 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00021 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00022 SOFTWARE.
00023 */
00024
00032 #ifndef INCLUDE_CTTHREAD_HPP_
00033 #define INCLUDE_CTTHREAD_HPP_
00034
00035 #include "core.hpp"
00036
00037 #include <thread>
00038
00051 class CtThread {
00052 public:
00060     EXPORTED_API static void sleepFor(CtUInt64 time);
00061
00062 protected:
00068     EXPORTED_API CtThread();
00069
00075     EXPORTED_API virtual ~CtThread();
00076
00085     EXPORTED_API CtBool isRunning();
00086
00095     EXPORTED_API void start();
00096
00104     EXPORTED_API void stop();
00105
00112     EXPORTED_API virtual void join();
00113
00120     EXPORTED_API virtual void loop() = 0;
00121
00122 protected:
00130     void setRunning(CtBool running);
00131
00132 private:
00136     void run();
00137
00138 private:
00139     CtAtomic<CtBool> m_running;
00140     std::thread m_thread;
00141 };
00142
00143 #endif //INCLUDE_CTTHREAD_HPP_
```

9.43 include/threading/CtWorker.hpp File Reference

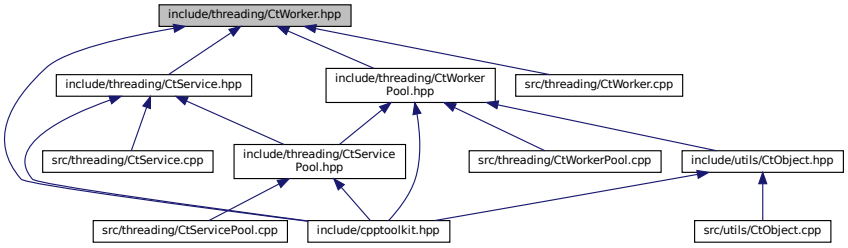
CtWorker class header file.

```
#include "core.hpp"
#include "threading/CtTask.hpp"
#include <thread>
#include <functional>
```

Include dependency graph for CtWorker.hpp:



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



Classes

- class CtWorker

  The CtWorker class provides a mechanism for executing tasks asynchronously in a separate thread.

9.43.1 Detailed Description

CtWorker class header file.

Date

18-01-2024

Definition in file CtWorker.hpp.

9.44 CtWorker.hpp

```
00001 /*
00002 MIT License
00003
00004 Copyright (c) 2024 Mouzenidis Panagiotis
00005
00006 Permission is hereby granted, free of charge, to any person obtaining a copy
00007 of this software and associated documentation files (the "Software"), to deal
00008 in the Software without restriction, including without limitation the rights
00009 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00010 copies of the Software, and to permit persons to whom the Software is
00011 furnished to do so, subject to the following conditions:
00012
00013 The above copyright notice and this permission notice shall be included in all
00014 copies or substantial portions of the Software.
00015
00016 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00017 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00018 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00019 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00020 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00021 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00022 SOFTWARE.
00023 */
00024
00032 #ifndef INCLUDE_CTWORKER_HPP_
00033 #define INCLUDE_CTWORKER_HPP_
00034
00035 #include "core.hpp"
00036
00037 #include "threading/CtTask.hpp"
00038
00039 #include <thread>
```

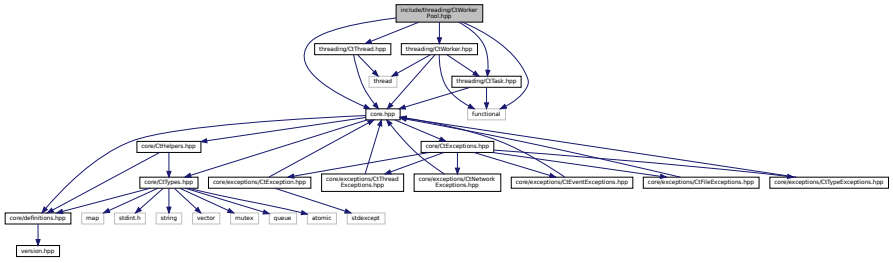
```
00040 #include <functional>
00041
00064 class CtWorker {
00065 public:
00072     EXPORTED_API explicit CtWorker();
00073
00080     EXPORTED_API ~CtWorker();
00081
00089     EXPORTED_API CtBool isRunning();
00090
00098     EXPORTED_API void runTask();
00099
00106     EXPORTED_API void joinTask();
00107
00119     EXPORTED_API void setTask(const CtTask& task, std::function<void()> callback = [](){});
00120
00130     template <typename F, typename... FArgs>
00131     EXPORTED_API void setTaskFunc(const F&& func, FArgs&&... fargs);
00132
00133 private:
00140     void alreadyRunningCheck();
00141
00148     void setRunning(CtBool running);
00149
00150 private:
00151     CtTask m_task;
00152     CtAtomic<CtBool> m_running;
00153     std::thread m_thread;
00154     std::function<void()> m_callback;
00155 };
00156
00157 template <typename F, typename... FArgs>
00158 void CtWorker::setTaskFunc(const F&& func, FArgs&&... fargs) {
00159     CtTask s_task;
00160     s_task.setTaskFunc(std::bind(func, std::forward<FArgs>(fargs)...));
00161     setTask(s_task);
00162 };
00163
00164 #endif //INCLUDE_CTWORKER_HPP_
```

9.45 include/threading/CtWorkerPool.hpp File Reference

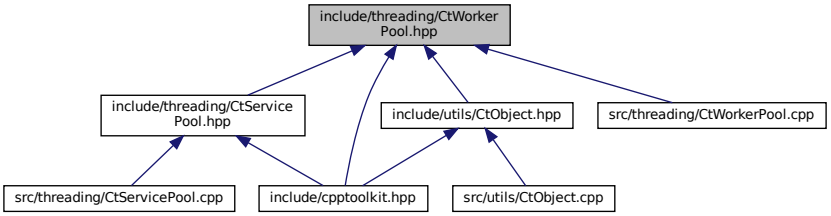
CtWorkerPool class header file.

```
#include "core.hpp"
#include "threading/CtWorker.hpp"
#include "threading/CtThread.hpp"
#include "threading/CtTask.hpp"
#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.

9.45.1 Detailed Description

CtWorkerPool class header file.

Date

18-01-2024

Definition in file CtWorkerPool.hpp.

9.46 CtWorkerPool.hpp

```
00001 /*
00002 MIT License
00003
00004 Copyright (c) 2024 Mouzenidis Panagiotis
00005
00006 Permission is hereby granted, free of charge, to any person obtaining a copy
00007 of this software and associated documentation files (the "Software"), to deal
00008 in the Software without restriction, including without limitation the rights
00009 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00010 copies of the Software, and to permit persons to whom the Software is
00011 furnished to do so, subject to the following conditions:
00012
00013 The above copyright notice and this permission notice shall be included in all
00014 copies or substantial portions of the Software.
00015
00016 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00017 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00018 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00019 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00020 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00021 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00022 SOFTWARE.
00023 */
00024
00032 #ifndef INCLUDE_CTWORKERPOOL_HPP_
00033 #define INCLUDE_CTWORKERPOOL_HPP_
00034
00035 #include "core.hpp"
00036
00037 #include "threading/CtWorker.hpp"
00038 #include "threading/CtThread.hpp"
00039 #include "threading/CtTask.hpp"
```

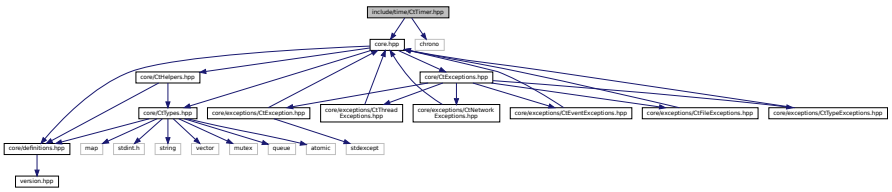
```
00040
00041 #include <functional>
00042
00069 class CtWorkerPool : private CtThread {
00070 public:
00078     EXPORTED_API explicit CtWorkerPool(CtUInt32 nworkers);
00079
00087     EXPORTED_API ~CtWorkerPool();
00088
00096     EXPORTED_API void addTask(const CtTask& task);
00097
00106     template <typename F, typename... FArgs>
00107     EXPORTED_API void addTask(const F&& func, FArgs&&... fargs);
00108
00115     EXPORTED_API void join() override;
00116
00117 private:
00126     void assignTask(CtUInt32 idx);
00127
00134     void free();
00135
00143     void loop() override;
00144
00145 private:
00146     CtUInt32 m_nworkers;
00147     CtVector<std::unique_ptr<CtWorker>> m_workers;
00148     CtQueue<CtTask> m_tasks;
00149     CtQueue<CtUInt32> m_available_workers_idxs;
00150     CtMutex m_mtx_control;
00151     CtAtomic<CtUInt32> m_active_tasks;
00152     CtAtomic<CtUInt32> m_queued_tasks;
00153     CtWorker m_taskAssigner;
00154 };
00155
00156 template <typename F, typename... FArgs>
00157 void CtWorkerPool::addTask(const F&& func, FArgs&&... fargs) {
00158     CtTask s_task;
00159     s_task.setTaskFunc(std::bind(func, std::forward<FArgs>(fargs)...));
00160     addTask(s_task);
00161 };
00162
00163 #endif //INCLUDE_CTWORKERPOOL_HPP_
```

9.47 include/time/CtTimer.hpp File Reference

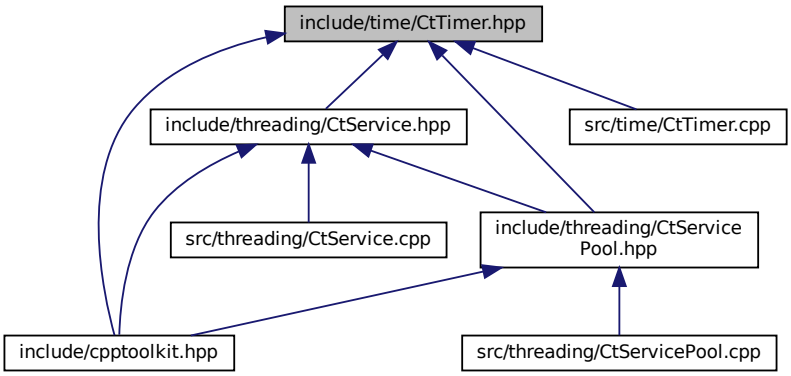
CtTimer class header file.

```
#include "core.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.

9.47.1 Detailed Description

CtTimer class header file.

Date

18-01-2024

Definition in file CtTimer.hpp.

9.48 CtTimer.hpp

```
00001 /*
00002 MIT License
00003
00004 Copyright (c) 2024 Mouzenidis Panagiotis
00005
00006 Permission is hereby granted, free of charge, to any person obtaining a copy
00007 of this software and associated documentation files (the "Software"), to deal
00008 in the Software without restriction, including without limitation the rights
00009 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00010 copies of the Software, and to permit persons to whom the Software is
00011 furnished to do so, subject to the following conditions:
00012
00013 The above copyright notice and this permission notice shall be included in all
00014 copies or substantial portions of the Software.
00015
00016 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00017 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00018 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00019 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00020 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00021 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00022 SOFTWARE.
00023 */
```

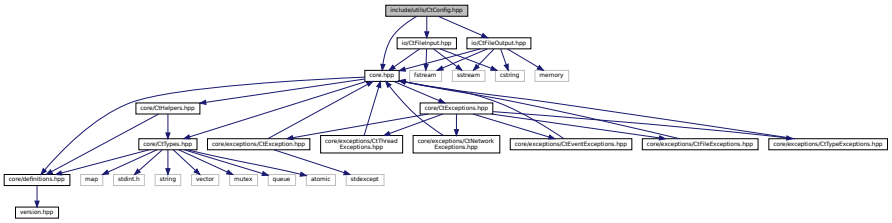
```
00024
00032 #ifndef INCLUDE_CTTIMER_HPP_
00033 #define INCLUDE_CTTIMER_HPP_
00034
00035 #include "core.hpp"
00036
00037 #include <chrono>
00038
00055 class CtTimer {
00056 public:
00062     EXPORTED_API CtTimer();
00063
00069     EXPORTED_API ~CtTimer();
00070
00076     EXPORTED_API void tic();
00077
00085     EXPORTED_API CtUInt64 toc();
00086
00094     EXPORTED_API static CtUInt64 current();
00095
00096 private:
00097     CtUInt64 m_reference;
00098 };
00099
00100 #endif //INCLUDE_CTTIMER_HPP_
```

9.49 include/utils/CtConfig.hpp File Reference

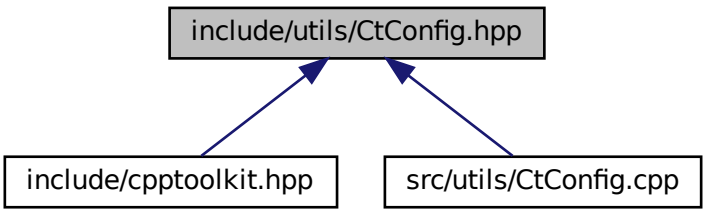
CtConfig class header file.

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

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.

9.49.1 Detailed Description

CtConfig class header file.

Date

10-03-2024

Definition in file CtConfig.hpp.

9.50 CtConfig.hpp

```
00001 /*
00002 MIT License
00003
00004 Copyright (c) 2024 Mouzenidis Panagiotis
00005
00006 Permission is hereby granted, free of charge, to any person obtaining a copy
00007 of this software and associated documentation files (the "Software"), to deal
00008 in the Software without restriction, including without limitation the rights
00009 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00010 copies of the Software, and to permit persons to whom the Software is
00011 furnished to do so, subject to the following conditions:
00012
00013 The above copyright notice and this permission notice shall be included in all
00014 copies or substantial portions of the Software.
00015
00016 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00017 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00018 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00019 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00020 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00021 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00022 SOFTWARE.
00023 */
00024
00032 #ifndef INCLUDE_CTCONFIG_HPP_
00033 #define INCLUDE_CTCONFIG_HPP_
00034
00035 #include "core.hpp"
00036
00037 #include "io/CtFileOutput.hpp"
00038 #include "io/CtFileInput.hpp"
00039
00049 class CtConfig {
00050 public:
00058     EXPORTED_API explicit CtConfig(const CtString& p_configFile);
00059
00065     EXPORTED_API ~CtConfig();
00066
00076     EXPORTED_API void read();
00077
00084     EXPORTED_API void write();
00085
00098     EXPORTED_API CtInt32 parseAsInt(const CtString& p_key);
00099
00112     EXPORTED_API CtUInt32 parseAsUInt(const CtString& p_key);
00113
00126     EXPORTED_API CtFloat parseAsFloat(const CtString& p_key);
00127
00140     EXPORTED_API CtDouble parseAsDouble(const CtString& p_key);
00141
00152     EXPORTED_API CtString parseAsString(const CtString& p_key);
00153
00163     EXPORTED_API void writeInt(const CtString& p_key, const CtInt32& p_value);
00164
00174     EXPORTED_API void writeUInt(const CtString& p_key, const CtUInt32& p_value);
```

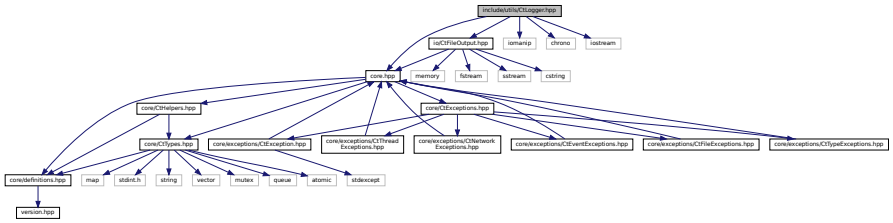
```
00175
00185     EXPORTED_API void writeFloat(const CtString& p_key, const CtFloat& p_value);
00186
00196     EXPORTED_API void writeDouble(const CtString& p_key, const CtDouble& p_value);
00197
00207     EXPORTED_API void writeString(const CtString& p_key, const CtString& p_value);
00208
00216     EXPORTED_API void reset();
00217
00218 private:
00229     CtString getValue(const CtString& p_key);
00230
00241     void parseLine(const CtString& p_line);
00242
00243 private:
00244     CtMutex m_mtx_control;
00245     CtFileInput* m_source;
00246     CtFileOutput* m_sink;
00247     CtString m_configFile;
00248     CtMap<CtString, CtString> m_configValues;
00249 };
00250
00251 #endif //INCLUDE_CTCONFIG_HPP_
```

9.51 include/utils/CtLogger.hpp File Reference

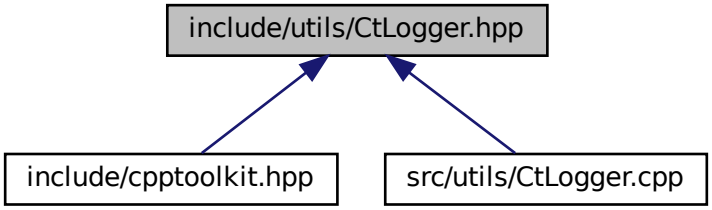
CtLogger class header file.

```
#include "core.hpp"
#include "io/CtFileOutput.hpp"
#include <iomanip>
#include <chrono>
#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.

9.51.1 Detailed Description

CtLogger class header file.

Date

10-03-2024

Definition in file CtLogger.hpp.

9.52 CtLogger.hpp

```
00001 /*
00002 MIT License
00003
00004 Copyright (c) 2024 Mouzenidis Panagiotis
00005
00006 Permission is hereby granted, free of charge, to any person obtaining a copy
00007 of this software and associated documentation files (the "Software"), to deal
00008 in the Software without restriction, including without limitation the rights
00009 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00010 copies of the Software, and to permit persons to whom the Software is
00011 furnished to do so, subject to the following conditions:
00012
00013 The above copyright notice and this permission notice shall be included in all
00014 copies or substantial portions of the Software.
00015
00016 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00017 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00018 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00019 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00020 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00021 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00022 SOFTWARE.
00023 */
00024
00032 #ifndef INCLUDE_CTLOGGER_HPP_
00033 #define INCLUDE_CTLOGGER_HPP_
00034
00035 #include "core.hpp"
00036
00037 #include "io/CtFileOutput.hpp"
00038
00039 #include <iomanip>
00040 #include <chrono>
00041 #include <iostream>
00042
00051 class CtLogger {
00052 public:
00058     enum class Level { DEBUG, INFO, WARNING, ERROR, CRITICAL };
00059
00068     EXPORTED_API explicit CtLogger(CtLogger::Level level = CtLogger::Level::DEBUG, const CtString&
      componentName = "");
00069
00075     EXPORTED_API ~CtLogger();
00076
00084     EXPORTED_API void log_debug(const CtString& message);
00085
00093     EXPORTED_API void log_info(const CtString& message);
00094
00102     EXPORTED_API void log_warning(const CtString& message);
00103
00111     EXPORTED_API void log_error(const CtString& message);
00112
00120     EXPORTED_API void log_critical(const CtString& message);
00121
00122 private:
```

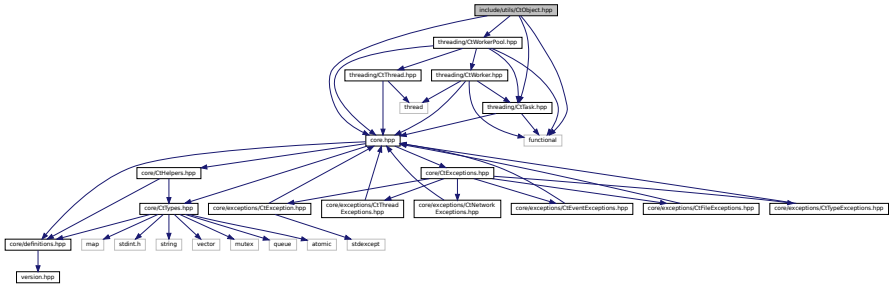
```
00133     void log(CtLogger::Level level, const CtString& message);
00134
00143     static const CtString levelToString(CtLogger::Level level);
00144
00155     static const CtString generateLoggerMsg(CtLogger::Level level, const CtString& component_name,
      const CtString& message);
00156
00157 private:
00158     CtMutex m_mtx_control;
00159     CtLogger::Level m_level;
00160     CtString m_componentName;
00161 };
00162
00163 #endif //INCLUDE_CTLOGGER_HPP_
```

9.53 include/utils/CtObject.hpp File Reference

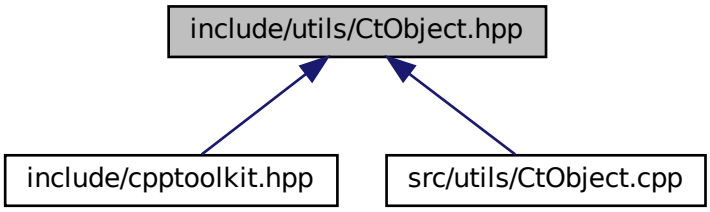
CtObject class header file.

```
#include "core.hpp"
#include "threading/CtTask.hpp"
#include "threading/CtWorkerPool.hpp"
#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.

9.53.1 Detailed Description

CtObject class header file.

Date

02-02-2024

Definition in file CtObject.hpp.

9.54 CtObject.hpp

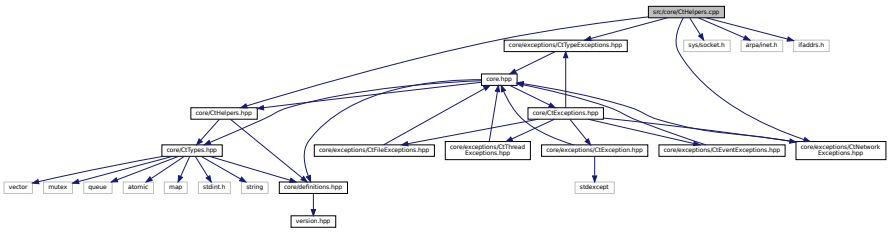
```
00001 /*
00002 MIT License
00003
00004 Copyright (c) 2024 Mouzenidis Panagiotis
00005
00006 Permission is hereby granted, free of charge, to any person obtaining a copy
00007 of this software and associated documentation files (the "Software"), to deal
00008 in the Software without restriction, including without limitation the rights
00009 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00010 copies of the Software, and to permit persons to whom the Software is
00011 furnished to do so, subject to the following conditions:
00012
00013 The above copyright notice and this permission notice shall be included in all
00014 copies or substantial portions of the Software.
00015
00016 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00017 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00018 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00019 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00020 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00021 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00022 SOFTWARE.
00023 */
00024
00032 #ifndef INCLUDE_CTOBJECT_HPP_
00033 #define INCLUDE_CTOBJECT_HPP_
00034
00035 #include "core.hpp"
00036
00037 #include "threading/CtTask.hpp"
00038 #include "threading/CtWorkerPool.hpp"
00039
00040 #include <functional>
00041
00059 class CtObject {
00060 public:
00077     template <typename F, typename... FArgs>
00078     EXPORTED_API static void connectEvent(CtObject* p_obj, CtUInt32 p_eventCode, F&& func, FArgs&&...
      fargs);
00079
00092     EXPORTED_API static void connectEvent(CtObject* p_obj, CtUInt32 p_eventCode, CtTask& p_task);
00093
00109     template <typename F, typename... FArgs>
00110     EXPORTED_API void connectEvent(CtUInt32 p_eventCode, F&& func, FArgs&&... fargs);
00111
00123     EXPORTED_API void connectEvent(CtUInt32 p_eventCode, CtTask& p_task);
00124
00133     EXPORTED_API void waitPendingEvents();
00134
00135 protected:
00141     EXPORTED_API CtObject();
00142
00149     EXPORTED_API ~CtObject();
00150
00161     EXPORTED_API void triggerEvent(CtUInt32 p_eventCode);
00162
00173     EXPORTED_API void registerEvent(CtUInt32 p_eventCode);
00174
00175 private:
00186     EXPORTED_API CtBool hasEvent(CtUInt32 p_eventCode);
00187
00188 private:
00189     CtMutex m_mtx_control;
00190     CtVector<CtUInt32> m_events;
00191     CtMultiMap<CtUInt32, CtTask> m_triggers;
```

```
00192     CtWorkerPool m_pool;
00193 };
00194
00195 template <typename F, typename... FArgs>
00196 void CtObject::connectEvent(CtObject* p_obj, CtUInt32 p_eventCode, F&& func, FArgs&&... fargs) {
00197     CtTask s_task;
00198     s_task.setTaskFunc(std::bind(func, std::forward<FArgs>(fargs)...));
00199     p_obj->connectEvent(p_eventCode, s_task);
00200 };
00201
00202 template <typename F, typename... FArgs>
00203 void CtObject::connectEvent(CtUInt32 p_eventCode, F&& func, FArgs&&... fargs) {
00204     CtTask s_task;
00205     s_task.setTaskFunc(std::bind(func, std::forward<FArgs>(fargs)...));
00206     connectEvent(p_eventCode, s_task);
00207 };
00208
00209 #endif //INCLUDE_CTOBJECT_HPP_
```

9.55 src/core/CtHelpers.cpp File Reference

```
#include "core/CtHelpers.hpp"
#include "core/exceptions/CtTypeExceptions.hpp"
#include "core/exceptions/CtNetworkExceptions.hpp"
#include <sys/socket.h>
#include <arpa/inet.h>
#include <ifaddrs.h>
```

Include dependency graph for CtHelpers.cpp:



9.55.1 Detailed Description

Date

31-01-2025

Definition in file CtHelpers.cpp.

9.56 CtHelpers.cpp

```
00001 /*
00002 MIT License
00003
00004 Copyright (c) 2024 Mouzenidis Panagiotis
00005
00006 Permission is hereby granted, free of charge, to any person obtaining a copy
00007 of this software and associated documentation files (the "Software"), to deal
00008 in the Software without restriction, including without limitation the rights
00009 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00010 copies of the Software, and to permit persons to whom the Software is
00011 furnished to do so, subject to the following conditions:
00012
```

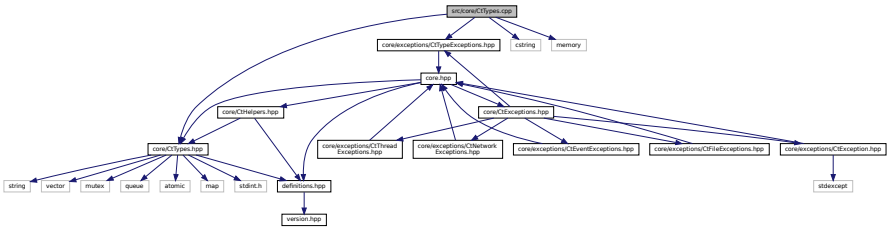
```
00013 The above copyright notice and this permission notice shall be included in all
00014 copies or substantial portions of the Software.
00015
00016 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00017 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00018 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00019 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00020 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00021 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00022 SOFTWARE.
00023 */
00024
00032 #include "core/CtHelpers.hpp"
00033
00034 #include "core/exceptions/CtTypeExceptions.hpp"
00035 #include "core/exceptions/CtNetworkExceptions.hpp"
00036
00037 #include <sys/socket.h>
00038 #include <arpa/inet.h>
00039 #include <ifaddrs.h>
00040
00041 void CtStringHelpers::split(const CtString& p_string, CtChar p_delimiter, CtVector<CtString>
       *p_result) {
00042     CtString::size_type s_start = 0;
00043     auto s_end = p_string.find(p_delimiter);
00044
00045     while (s_end != CtString::npos) {
00046         p_result->push_back(trim(p_string.substr(s_start, s_end - s_start)));
00047         s_start = s_end + 1;
00048         s_end = p_string.find(p_delimiter, s_start);
00049     }
00050
00051     p_result->push_back(trim(p_string.substr(s_start)));
00052 }
00053
00054
00055 CtString CtStringHelpers::trim(const CtString& p_string) {
00056     auto s_start = p_string.find_first_not_of(" \t\n");
00057     auto s_end = p_string.find_last_not_of(" \t\n");
00058     return ((s_start == CtString::npos) ? "" : p_string.substr(s_start, s_end - s_start + 1));
00059 }
00060
00061 CtDouble CtStringHelpers::StrToDouble(const CtString& p_string) {
00062     CtDouble parsed_value;
00063     try {
00064         parsed_value = stod(p_string);
00065     } catch (...) {
00066         throw CtTypeParseError(p_string + CtString(" can not be parsed as CtDouble."));
00067     }
00068     return parsed_value;
00069 }
00070
00071 CtFloat CtStringHelpers::StrToFloat(const CtString& p_string) {
00072     CtDouble parsed_value;
00073     try {
00074         parsed_value = stof(p_string);
00075     } catch (...) {
00076         throw CtTypeParseError(p_string + CtString(" can not be parsed as CtFloat."));
00077     }
00078     return parsed_value;
00079 }
00080
00081 CtUInt32 CtStringHelpers::StrToUInt(const CtString& p_string) {
00082     CtUInt32 parsed_value;
00083     try {
00084         parsed_value = stoul(p_string);
00085     } catch (...) {
00086         throw CtTypeParseError(p_string + CtString(" can not be parsed as uint."));
00087     }
00088     return parsed_value;
00089 }
00090
00091 CtInt32 CtStringHelpers::StrToInt(const CtString& p_string) {
00092     CtInt32 parsed_value;
00093     try {
00094         parsed_value = stoi(p_string);
00095     } catch (...) {
00096         throw CtTypeParseError(p_string + CtString(" can not be parsed as int."));
00097     }
00098     return parsed_value;
00099 }
00100
00101 void CtSocketHelpers::setSocketTimeout(CtInt32 p_socketTimeout) {
00102     CtSocketHelpers::socketTimeout = p_socketTimeout;
00103 }
00104
00105 CtVector<CtString> CtSocketHelpers::getInterfaces() {
```

```
00106     struct ifaddrs *s_ifaddr, *s_ifa;
00107     CtVector<CtString> s_interfaces;
00108
00109     if (getifaddrs(&s_ifaddr) == -1) {
00110         throw CtSocketError("Cannot get interfaces.");
00111     }
00112
00113     for (s_ifa = s_ifaddr; s_ifa != nullptr; s_ifa = s_ifa->ifa_next) {
00114         s_interfaces.push_back(CtString(s_ifa->ifa_name));
00115     }
00116     return s_interfaces;
00117 }
00118
00119 CtString CtSocketHelpers::interfaceToAddress(const CtString& p_ifName) {
00120     struct ifaddrs *ifaddr, *ifa;
00121
00122     if (getifaddrs(&ifaddr) == -1) {
00123         throw CtSocketError("Cannot get interfaces.");
00124     }
00125
00126     for (ifa = ifaddr; ifa != nullptr; ifa = ifa->ifa_next) {
00127         if (CtString(ifa->ifa_name) == p_ifName) {
00128             if (ifa->ifa_addr == nullptr) {
00129                 freeifaddrs(ifaddr);
00130                 throw CtSocketError("Not valid interface entered.");
00131             }
00132
00133             if (ifa->ifa_addr->sa_family == AF_INET) {
00134                 CtChar ipBuffer[INET_ADDRSTRLEN];
00135                 sockaddr_in* sockAddr = reinterpret_cast<sockaddr_in*>(ifa->ifa_addr);
00136
00137                 if (inet_ntop(AF_INET, &(sockAddr->sin_addr), ipBuffer, INET_ADDRSTRLEN) == nullptr) {
00138                     freeifaddrs(ifaddr);
00139                     throw CtSocketError("Failed to convert IPv4 address.");
00140                 }
00141
00142                 freeifaddrs(ifaddr);
00143                 return ipBuffer;
00144             }
00145         }
00146     }
00147
00148     freeifaddrs(ifaddr);
00149     throw CtSocketError("Not valid interface found.");
00150 }
00151
00152 CtUInt32 CtSocketHelpers::getAddressAsUInt(const CtString& p_addr) {
00153     CtUInt32 result = inet_addr(p_addr.c_str());
00154
00155     if (result == INADDR_NONE) {
00156         throw CtSocketError("Invalid address given.");
00157     }
00158
00159     return result;
00160 };
00161
00162 CtString CtSocketHelpers::getAddressAsString(CtUInt32 p_addr) {
00163     CtChar result[INET_ADDRSTRLEN];
00164
00165     if (inet_ntop(AF_INET, &p_addr, result, INET_ADDRSTRLEN) == nullptr) {
00166         throw CtSocketError("Failed to convert IPv4 address.");
00167     }
00168
00169     return CtString(result);
00170 };
```

9.57 src/core/CtTypes.cpp File Reference

```
#include "core/CtTypes.hpp"
#include "core/exceptions/CtTypeExceptions.hpp"
#include <cstring>
#include <memory>
```

Include dependency graph for CtTypes.cpp:



9.57.1 Detailed Description

Date

01-02-2025

Definition in file CtTypes.cpp.

9.58 CtTypes.cpp

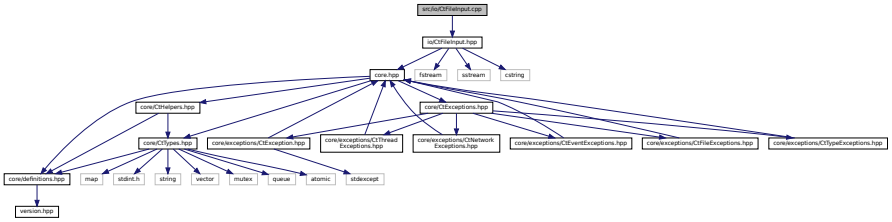
```
00001 /*
00002 MIT License
00003
00004 Copyright (c) 2024 Mouzenidis Panagiotis
00005
00006 Permission is hereby granted, free of charge, to any person obtaining a copy
00007 of this software and associated documentation files (the "Software"), to deal
00008 in the Software without restriction, including without limitation the rights
00009 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00010 copies of the Software, and to permit persons to whom the Software is
00011 furnished to do so, subject to the following conditions:
00012
00013 The above copyright notice and this permission notice shall be included in all
00014 copies or substantial portions of the Software.
00015
00016 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00017 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00018 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00019 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00020 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00021 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00022 SOFTWARE.
00023 */
00024
00032 #include "core/CtTypes.hpp"
00033
00034 #include "core/exceptions/CtTypeExceptions.hpp"
00035
00036 #include <cstring>
00037 #include <memory>
00038
00039 CtRawData::CtRawData(CtUInt32 p_size) : m_maxSize(p_size) {
00040     m_data = new CtUInt8[m_maxSize];
00041     m_size = 0;
00042 };
00043
00044 CtRawData::CtRawData(CtRawData& p_data) : m_maxSize(p_data.maxSize()) {
00045     m_data = new CtUInt8[m_maxSize];
00046     clone(p_data);
00047 };
00048
00049 CtRawData::~CtRawData() {
00050     delete[] m_data;
00051 }
00052
00053 void CtRawData::setNextByte(CtUInt8 p_data) {
00054     if (m_size >= m_maxSize) {
00055         throw CtOutOfRangeError("Data size is out of range.");
```

```
00056     }
00057     m_data[m_size++] = p_data;
00058 }
00059
00060 void CtRawData::setNextBytes(CtUInt8* p_data, CtUInt32 p_size) {
00061     if (m_size + p_size > m_maxSize) {
00062         throw CtOutOfRangeError("Data size is out of range.");
00063     }
00064     memcpy(&m_data[m_size], p_data, p_size);
00065     m_size += p_size;
00066 }
00067
00068 CtUInt8* CtRawData::getNLastBytes(CtUInt32 p_num) {
00069     if (p_num > m_size) {
00070         throw CtOutOfRangeError("Data size is out of range.");
00071     }
00072     return &m_data[m_size - p_num];
00073 }
00074
00075 void CtRawData::removeNLastBytes(CtUInt32 p_num) {
00076     if (p_num > m_size) {
00077         throw CtOutOfRangeError("Data size is out of range.");
00078     }
00079     m_size -= p_num;
00080 }
00081
00082 CtUInt32 CtRawData::size() {
00083     return m_size;
00084 }
00085
00086 CtUInt32 CtRawData::maxSize() {
00087     return m_maxSize;
00088 }
00089
00090 CtUInt8* CtRawData::get() {
00091     return m_data;
00092 }
00093
00094 void CtRawData::clone(const CtUInt8* p_data, CtUInt32 p_size) {
00095     if (p_size > m_maxSize) {
00096         throw CtOutOfRangeError("Data size is out of range.");
00097     }
00098     m_size = p_size;
00099     memcpy(m_data, p_data, p_size);
00100 }
00101
00102 void CtRawData::clone(CtRawData& p_data) {
00103     if (p_data.size() > m_maxSize) {
00104         throw CtOutOfRangeError("Data size is out of range.");
00105     }
00106     m_size = p_data.size();
00107     memcpy(m_data, p_data.get(), p_data.size());
00108 }
00109
00110 void CtRawData::reset() {
00111     m_size = 0;
00112 }
00113
00114 CtRawData& CtRawData::operator=(CtRawData& other) {
00115     if (this != &other) {
00116         clone(other);
00117     }
00118     return *this;
00119 }
```

9.59 src/io/CtFileInput.cpp File Reference

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

Include dependency graph for CtFileInput.cpp:



9.59.1 Detailed Description

Date

08-03-2024

Definition in file CtFileInput.cpp.

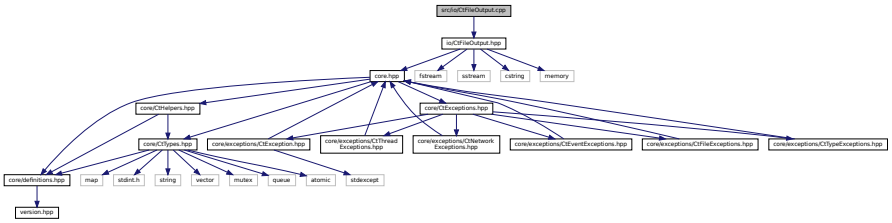
9.60 CtFileInput.cpp

```
00001 /*
00002 MIT License
00003
00004 Copyright (c) 2024 Mouzenidis Panagiotis
00005
00006 Permission is hereby granted, free of charge, to any person obtaining a copy
00007 of this software and associated documentation files (the "Software"), to deal
00008 in the Software without restriction, including without limitation the rights
00009 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00010 copies of the Software, and to permit persons to whom the Software is
00011 furnished to do so, subject to the following conditions:
00012
00013 The above copyright notice and this permission notice shall be included in all
00014 copies or substantial portions of the Software.
00015
00016 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00017 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00018 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00019 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00020 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00021 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00022 SOFTWARE.
00023 */
00024
00032 #include "io/CtFileInput.hpp"
00033
00034 CtFileInput::CtFileInput(const CtString& p_fileName) {
00035     m_delim = nullptr;
00036     m_delim_size = 0;
00037     m_file.open(p_fileName, std::ofstream::in);
00038     if (!m_file.is_open()) {
00039         throw CtFileReadError("File cannot open.");
00040     }
00041 }
00042
00043 CtFileInput::~CtFileInput() {
00044     if (m_file.is_open()) {
00045         m_file.close();
00046     }
00047     if (m_delim != nullptr) {
00048         delete[] m_delim;
00049     }
00050 }
00051
00052 void CtFileInput::setDelimiter(const CtChar* p_delim, CtUInt8 p_delim_size) {
00053     if (p_delim_size > 0 && p_delim != nullptr) {
00054         m_delim_size = p_delim_size;
00055         m_delim = new CtChar[m_delim_size];
00056         memcpy(m_delim, p_delim, m_delim_size);
```

```
00057     } else {
00058         m_delim_size = 0;
00059         if (m_delim != nullptr) {
00060             delete[] m_delim;
00061         }
00062     }
00063 }
00064
00065 CtBool CtFileInput::read(CtRawData* p_data) {
00066     CtBool s_res = CT_FALSE;
00067
00068     if (m_file.is_open()) {
00069         CtChar next_char;
00070         CtUInt8* delim_ptr = nullptr;
00071         p_data->reset();
00072
00073         while (m_file.get(next_char)) {
00074             try {
00075                 p_data->setNextByte(next_char);
00076             } catch (const CtOutOfRangeError& e) {
00077                 m_file.seekg(-(m_delim_size+1), std::ios::cur);
00078                 p_data->removeNLastBytes(m_delim_size);
00079                 break;
00080             }
00081
00082             if (m_delim != nullptr && p_data->size() >= m_delim_size) {
00083                 delim_ptr = p_data->getNLastBytes(m_delim_size);
00084
00085                 if (memcmp(delim_ptr, m_delim, m_delim_size) == 0) {
00086                     p_data->removeNLastBytes(m_delim_size);
00087                     break;
00088                 }
00089             }
00090         }
00091
00092         if (p_data->size() > 0) {
00093             s_res = CT_TRUE;
00094         } else if (m_file.eof()) {
00095             s_res = CT_FALSE;
00096         }
00097     } else {
00098         throw CtFileReadError("File is not open.");
00099     }
00100
00101     return s_res;
00102 }
```

9.61 src/io/CtFileOutput.cpp File Reference

`#include "io/CtFileOutput.hpp"`
Include dependency graph for CtFileOutput.cpp:



9.61.1 Detailed Description

Date

09-03-2024

Definition in file CtFileOutput.cpp.

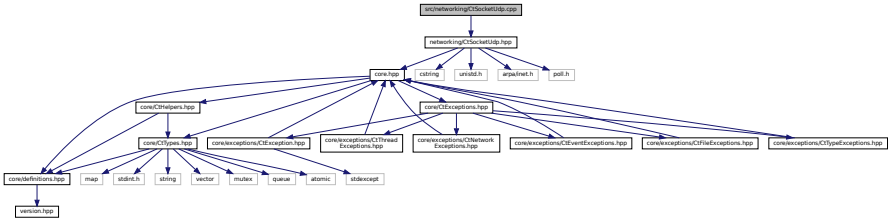
9.62 CtFileOutput.cpp

```
00001 /*
00002 MIT License
00003
00004 Copyright (c) 2024 Mouzenidis Panagiotis
00005
00006 Permission is hereby granted, free of charge, to any person obtaining a copy
00007 of this software and associated documentation files (the "Software"), to deal
00008 in the Software without restriction, including without limitation the rights
00009 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00010 copies of the Software, and to permit persons to whom the Software is
00011 furnished to do so, subject to the following conditions:
00012
00013 The above copyright notice and this permission notice shall be included in all
00014 copies or substantial portions of the Software.
00015
00016 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00017 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00018 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00019 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00020 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00021 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00022 SOFTWARE.
00023 */
00024
00032 #include "io/CtFileOutput.hpp"
00033
00034 CtFileOutput::CtFileOutput(const CtString& p_fileName, WriteMode p_mode) {
00035     m_delim_size = 0;
00036     switch (p_mode) {
00037         case WriteMode::Append:
00038             m_file.open(p_fileName, std::ios::out | std::ios::app);
00039             break;
00040             default:
00041         case WriteMode::Truncate:
00042             m_file.open(p_fileName, std::ios::out | std::ios::trunc);
00043             break;
00044     }
00045     if (!m_file.is_open()) {
00046         throw CtFileWriteError("File cannot open.");
00047     }
00048 }
00049
00050 CtFileOutput::~CtFileOutput() {
00051     if (m_file.is_open()) {
00052         m_file.close();
00053     }
00054 }
00055
00056 void CtFileOutput::setDelimiter(const CtChar* p_delim, CtUInt8 p_delim_size) {
00057     if (p_delim_size > 0 && p_delim != nullptr) {
00058         m_delim_size = p_delim_size;
00059         m_delim.reset();
00060         m_delim = std::make_unique<CtChar[]>(m_delim_size);
00061         memcpy(m_delim.get(), p_delim, m_delim_size);
00062     } else {
00063         m_delim_size = 0;
00064         m_delim.reset();
00065     }
00066 }
00067
00068 void CtFileOutput::write(CtRawData* p_data) {
00069     if (m_file.is_open()) {
00070         m_file.write((CtChar*)p_data->get(), p_data->size());
00071         if (m_delim_size > 0) {
00072             m_file.write(m_delim.get(), m_delim_size);
00073         }
00074     } else {
00075         throw CtFileWriteError("File is not open.");
00076     }
00077 }
00078
00079 void CtFileOutput::writePart(CtRawData* p_data) {
00080     if (m_file.is_open()) {
00081         m_file.write((CtChar*)p_data->get(), p_data->size());
00082     } else {
00083         throw CtFileWriteError("File is not open.");
00084     }
00085 }
```

9.63 src/networking/CtSocketUdp.cpp File Reference

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

Include dependency graph for CtSocketUdp.cpp:



9.63.1 Detailed Description

Date

18-01-2024

Definition in file CtSocketUdp.cpp.

9.64 CtSocketUdp.cpp

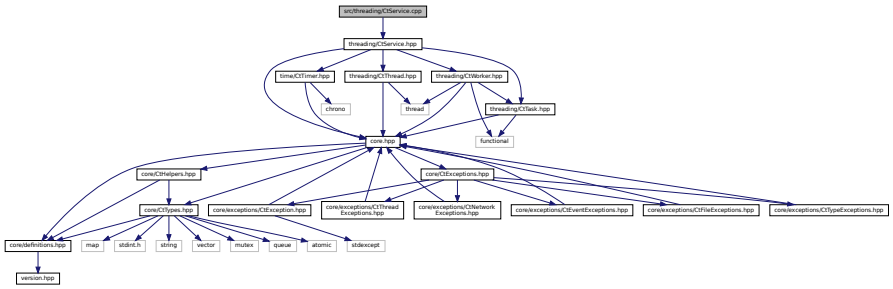
```
00001 /*
00002 MIT License
00003
00004 Copyright (c) 2024 Mouzenidis Panagiotis
00005
00006 Permission is hereby granted, free of charge, to any person obtaining a copy
00007 of this software and associated documentation files (the "Software"), to deal
00008 in the Software without restriction, including without limitation the rights
00009 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00010 copies of the Software, and to permit persons to whom the Software is
00011 furnished to do so, subject to the following conditions:
00012
00013 The above copyright notice and this permission notice shall be included in all
00014 copies or substantial portions of the Software.
00015
00016 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00017 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00018 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00019 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00020 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00021 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00022 SOFTWARE.
00023 */
00024
00032 #include "networking/CtSocketUdp.hpp"
00033
00034 CtSocketUdp::CtSocketUdp() {
00035     m_addrType = AF_INET;
00036     m_port = 0;
00037     m_socket = socket(m_addrType, SOCK_DGRAM, IPPROTO_UDP);
00038     if (m_socket == -1) {
00039         throw CtSocketError("Socket cannot be assigned.");
00040     }
00041     m_pollin_sockets[0].fd = m_socket;
00042     m_pollin_sockets[0].events = POLLIN;
00043
00044     m_pollout_sockets[0].fd = m_socket;
00045     m_pollout_sockets[0].events = POLLOUT;
00046 }
00047
00048 CtSocketUdp::~CtSocketUdp() {
00049     close(m_socket);
00050 }
```

```
00051
00052 void CtSocketUdp::setSub(const CtString& p_interfaceName, CtUInt16 p_port) {
00053     memset(&m_subAddress, 0, sizeof(m_subAddress));
00054     m_subAddress.sin_family = m_addrType;
00055     m_subAddress.sin_addr.s_addr =
       CtSocketHelpers::getAddressAsUInt(CtSocketHelpers::interfaceToAddress(p_interfaceName));
00056     m_subAddress.sin_port = htons(p_port);
00057
00058     if (bind(m_socket, (struct sockaddr*)&m_subAddress, sizeof(m_subAddress)) == -1) {
00059         throw CtSocketBindError(CtString("Socket bind to port ") + ToCtString(p_port) + CtString("
       failed."));
00060     }
00061 }
00062
00063 void CtSocketUdp::setPub(CtUInt16 p_port, const CtString& p_addr) {
00064     memset(&m_pubAddress, 0, sizeof(m_pubAddress));
00065     m_pubAddress.sin_family = m_addrType;
00066     m_pubAddress.sin_addr.s_addr = CtSocketHelpers::getAddressAsUInt(p_addr);
00067     m_pubAddress.sin_port = htons(p_port);
00068 }
00069
00070 CtBool CtSocketUdp::pollRead() {
00071     CtInt32 pollResult = poll(m_pollin_sockets, 1, CtSocketHelpers::socketTimeout);
00072
00073     if (pollResult < 0) {
00074         throw CtSocketPollError("Socket polling-in failed.");
00075     } else if (pollResult == 0) {
00076         return CT_FALSE;
00077     } else {
00078         return CT_TRUE;
00079     }
00080 }
00081
00082 CtBool CtSocketUdp::pollWrite() {
00083     CtInt32 pollResult = poll(m_pollout_sockets, 1, CtSocketHelpers::socketTimeout);
00084     if (pollResult < 0) {
00085         throw CtSocketPollError("Socket polling-out failed.");
00086     } else if (pollResult == 0) {
00087         return CT_FALSE;
00088     } else {
00089         return CT_TRUE;
00090     }
00091 }
00092
00093 void CtSocketUdp::send(CtUInt8* p_data, CtUInt32 p_size) {
00094     if (sendto(m_socket, p_data, p_size, MSG_DONTWAIT, (struct sockaddr*)&m_pubAddress,
       sizeof(m_pubAddress)) == -1) {
00095         throw CtSocketWriteError("Sending data via socket failed.");
00096     }
00097 }
00098
00099 void CtSocketUdp::send(CtRawData& p_message) {
00100     send(p_message.get(), p_message.size());
00101 }
00102
00103 void CtSocketUdp::receive(CtUInt8* p_data, CtUInt32 p_size, CtNetAddress* p_client) {
00104     sockaddr_in s_clientAddress_in;
00105     socklen_t s_clientAddressLength = sizeof(s_clientAddress_in);
00106     CtInt32 bytesRead = recvfrom(m_socket, p_data, p_size, MSG_DONTWAIT, (struct
       sockaddr*)&s_clientAddress_in, &s_clientAddressLength);
00107
00108     if (bytesRead == -1) {
00109         throw CtSocketReadError("Receiving data via socket failed.");
00110     }
00111
00112     if (p_client != nullptr) {
00113         p_client->addr =
       (CtString)CtSocketHelpers::getAddressAsString(*(CtUInt32*)(&s_clientAddress_in.sin_addr));
00114         p_client->port = s_clientAddress_in.sin_port;
00115     }
00116
00117     p_data[bytesRead] = '\0';
00118 }
00119
00120 void CtSocketUdp::receive(CtRawData* p_message, CtNetAddress* p_client) {
00121     CtUInt8* s_buffer = new CtUInt8[p_message->maxSize()];
00122     receive(s_buffer, p_message->maxSize(), p_client);
00123     p_message->clone(s_buffer, p_message->maxSize());
00124     delete[] s_buffer;
00125 }
```

9.65 src/threading/CtService.cpp File Reference

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

Include dependency graph for CtService.cpp:



9.65.1 Detailed Description

Date

18-01-2024

Definition in file CtService.cpp.

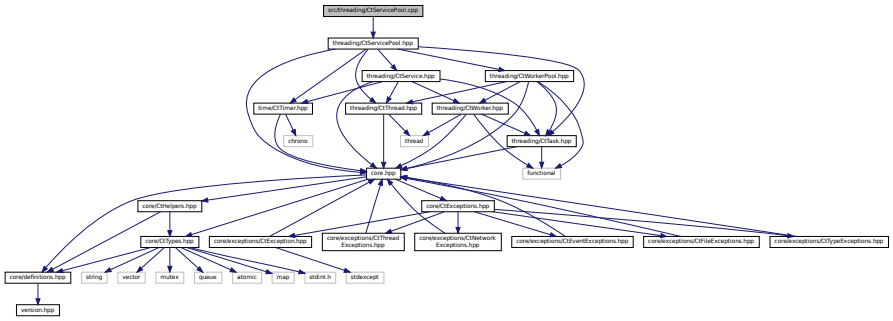
9.66 CtService.cpp

```
00001 /*
00002 MIT License
00003
00004 Copyright (c) 2024 Mouzenidis Panagiotis
00005
00006 Permission is hereby granted, free of charge, to any person obtaining a copy
00007 of this software and associated documentation files (the "Software"), to deal
00008 in the Software without restriction, including without limitation the rights
00009 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00010 copies of the Software, and to permit persons to whom the Software is
00011 furnished to do so, subject to the following conditions:
00012
00013 The above copyright notice and this permission notice shall be included in all
00014 copies or substantial portions of the Software.
00015
00016 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00017 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00018 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00019 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00020 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00021 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00022 SOFTWARE.
00023 */
00024
00032 #include "threading/CtService.hpp"
00033
00034 CtUInt32 CtService::m_slot_time = 10;
00035
00036 CtService::CtService(CtUInt64 nslots, const CtTask& task) : m_nslots(nslots), m_skip_ctr(0),
       m_exec_ctr(1) {
00037     m_worker.setTask(task);
00038 }
00039
00040 CtService::~CtService() {
00041     stopService();
00042 }
00043
00044 void CtService::runService() {
00045     try {
00046         start();
```

```
00047     } catch(const CtThreadError& e) {
00048         throw CtServiceError("Service is already running.");
00049     }
00050 }
00051
00052 void CtService::stopService() {
00053     stop();
00054     m_worker.joinTask();
00055 }
00056
00057 float CtService::getIntervalValidity() {
00058     return m_skip_ctr/(float)m_exec_ctr;
00059 }
00060
00061 void CtService::loop() {
00062     try {
00063         m_worker.runTask();
00064     } catch(const CtWorkerError& e) {
00065         /*
00066          * Service interval is smaller than the task runtime. The worker
00067          * is still running while it is time to run again.
00068          * Exception cannot be thrown at this point because this is
00069          * a thread running. The "controller" of this thread could not
00070          * catch this exception.
00071          */
00072         m_skip_ctr += 1;
00073     }
00074     m_exec_ctr += 1;
00075     CtThread::sleepFor(m_nslots*m_slot_time);
00076 }
```

9.67 src/threading/CtServicePool.cpp File Reference

`#include "threading/CtServicePool.hpp"`
Include dependency graph for CtServicePool.cpp:



9.67.1 Detailed Description

Date

18-01-2024

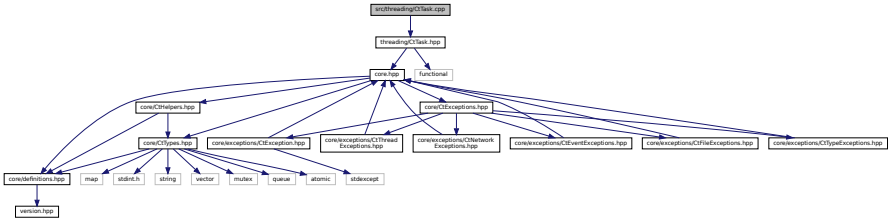
Definition in file CtServicePool.cpp.

9.68 CtServicePool.cpp

```
00001 /*
00002 MIT License
00003
00004 Copyright (c) 2024 Mouzenidis Panagiotis
00005
00006 Permission is hereby granted, free of charge, to any person obtaining a copy
00007 of this software and associated documentation files (the "Software"), to deal
00008 in the Software without restriction, including without limitation the rights
00009 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00010 copies of the Software, and to permit persons to whom the Software is
00011 furnished to do so, subject to the following conditions:
00012
00013 The above copyright notice and this permission notice shall be included in all
00014 copies or substantial portions of the Software.
00015
00016 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00017 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00018 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00019 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00020 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00021 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00022 SOFTWARE.
00023 */
00024
00032 #include "threading/CtServicePool.hpp"
00033
00034 CtServicePool::CtServicePool(CtUInt32 nworkers) : m_nworkers(nworkers), m_worker_pool(m_nworkers) {
00035     m_slot_cnt = 0;
00036     m_exec_time = 0;
00037 }
00038
00039 CtServicePool::~CtServicePool() {
00040     stop();
00041     m_worker_pool.join();
00042 }
00043
00044 void CtServicePool::addTask(CtUInt32 nslots, const CtString& id, CtTask& task) {
00045     std::scoped_lock lock(m_mtx_control);
00046     m_tasks.push_back({task, id, nslots});
00047 }
00048
00049 void CtServicePool::removeTask(const CtString& id) {
00050     std::scoped_lock lock(m_mtx_control);
00051     m_tasks.erase(std::remove_if(m_tasks.begin(), m_tasks.end(),
00052                                  [id](CtServicePack pack) {
00053                                      return pack.id.compare(id) == 0;
00054                                  }
00055                                  ), m_tasks.end());
00056 }
00057
00058 void CtServicePool::startServices() {
00059     try {
00060         start();
00061     } catch(const CtThreadError& e) {
00062     }
00063 }
00064
00065 void CtServicePool::shutdownServices() {
00066     stop();
00067     m_worker_pool.join();
00068 }
00069
00070 void CtServicePool::loop() {
00071     m_timer.tic();
00072     {
00073         std::scoped_lock lock(m_mtx_control);
00074         if (m_tasks.size() == 0) {
00075             return;
00076         } else {
00077             for (const CtServicePack& pack : m_tasks) {
00078                 if (m_slot_cnt % pack.nslots == 0) {
00079                     m_worker_pool.addTask(pack.task);
00080                 }
00081             }
00082         }
00083     }
00084     m_slot_cnt++;
00085     if (!isRunning()) return;
00086     m_exec_time = CtService::m_slot_time - m_timer.toc();
00087     if (m_exec_time > 0) {
00088         CtThread::sleepFor(m_exec_time);
00089     }
00090 }
```

9.69 src/threading/CtTask.cpp File Reference

`#include "threading/CtTask.hpp"`
Include dependency graph for CtTask.cpp:



9.69.1 Detailed Description

Date

18-01-2024

Definition in file CtTask.cpp.

9.70 CtTask.cpp

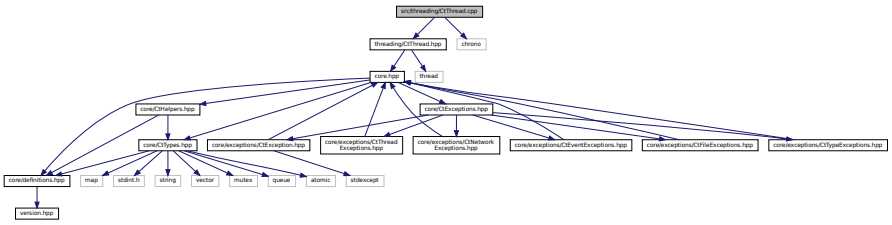
```
00001 /*
00002 MIT License
00003
00004 Copyright (c) 2024 Mouzenidis Panagiotis
00005
00006 Permission is hereby granted, free of charge, to any person obtaining a copy
00007 of this software and associated documentation files (the "Software"), to deal
00008 in the Software without restriction, including without limitation the rights
00009 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00010 copies of the Software, and to permit persons to whom the Software is
00011 furnished to do so, subject to the following conditions:
00012
00013 The above copyright notice and this permission notice shall be included in all
00014 copies or substantial portions of the Software.
00015
00016 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00017 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00018 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00019 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00020 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00021 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00022 SOFTWARE.
00023 */
00024
00032 #include "threading/CtTask.hpp"
00033
00034 CtTask::CtTask() : m_task([]{}), m_callback([]{}) {
00035 }
00036
00037 CtTask::CtTask(const CtTask& other) : m_task(other.m_task), m_callback(other.m_callback) {
00038 }
00039
00040 CtTask::~CtTask() {
00041 }
00042
00043 std::function<void()> CtTask::getTaskFunc() {
00044     return m_task;
00045 }
00046
00047 std::function<void()> CtTask::getCallbackFunc() {
00048     return m_callback;
00049 }
00050
```

```
00051 CtTask& CtTask::operator=(const CtTask& other) {
00052     if (this != &other) {
00053         m_task = other.m_task;
00054         m_callback = other.m_callback;
00055     }
00056     return *this;
00057 }
```

9.71 src/threading/CtThread.cpp File Reference

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

Include dependency graph for CtThread.cpp:



9.71.1 Detailed Description

Date

18-01-2024

Definition in file CtThread.cpp.

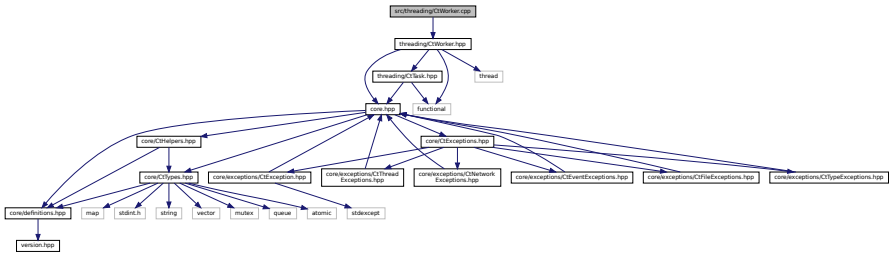
9.72 CtThread.cpp

```
00001 /*
00002 MIT License
00003
00004 Copyright (c) 2024 Mouzenidis Panagiotis
00005
00006 Permission is hereby granted, free of charge, to any person obtaining a copy
00007 of this software and associated documentation files (the "Software"), to deal
00008 in the Software without restriction, including without limitation the rights
00009 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00010 copies of the Software, and to permit persons to whom the Software is
00011 furnished to do so, subject to the following conditions:
00012
00013 The above copyright notice and this permission notice shall be included in all
00014 copies or substantial portions of the Software.
00015
00016 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00017 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00018 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00019 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00020 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00021 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00022 SOFTWARE.
00023 */
00024
00032 #include "threading/CtThread.hpp"
00033
00034 #include <chrono>
00035
00036 CtThread::CtThread() : m_running(CT_FALSE) {
00037 }
```

```
00038
00039 CtThread::~CtThread() {
00040     stop();
00041 }
00042
00043 void CtThread::run() {
00044     while(isRunning()) {
00045         loop();
00046     }
00047 }
00048
00049 void CtThread::start() {
00050     if (!isRunning()) {
00051         join();
00052         setRunning(CT_TRUE);
00053         m_thread = std::thread(&CtThread::run, this);
00054     } else {
00055         throw CtThreadError("Thread already running.");
00056     }
00057 }
00058
00059 void CtThread::stop() {
00060     setRunning(CT_FALSE);
00061     CtThread::join();
00062 }
00063
00064 void CtThread::join() {
00065     if (m_thread.joinable()) {
00066         m_thread.join();
00067     }
00068 }
00069
00070 CtBool CtThread::isRunning() {
00071     return m_running.load();
00072 }
00073
00074 void CtThread::setRunning(CtBool running) {
00075     m_running.store(running);
00076 }
00077
00078 void CtThread::sleepFor(CtUInt64 time) {
00079     std::this_thread::sleep_for(std::chrono::milliseconds(time));
00080 }
```

9.73 src/threading/CtWorker.cpp File Reference

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



9.73.1 Detailed Description

Date

18-01-2024

Definition in file CtWorker.cpp.

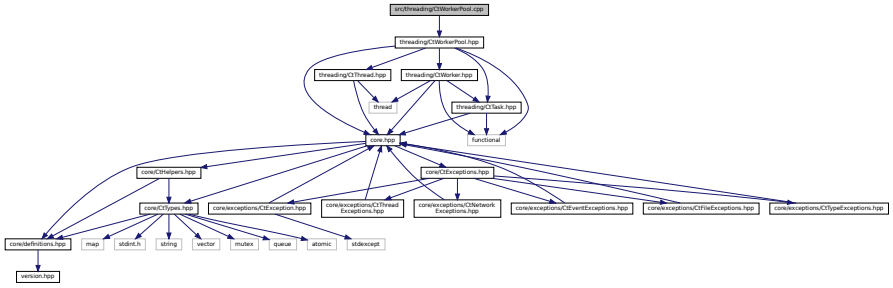
9.74 CtWorker.cpp

```
00001 /*
00002 MIT License
00003
00004 Copyright (c) 2024 Mouzenidis Panagiotis
00005
00006 Permission is hereby granted, free of charge, to any person obtaining a copy
00007 of this software and associated documentation files (the "Software"), to deal
00008 in the Software without restriction, including without limitation the rights
00009 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00010 copies of the Software, and to permit persons to whom the Software is
00011 furnished to do so, subject to the following conditions:
00012
00013 The above copyright notice and this permission notice shall be included in all
00014 copies or substantial portions of the Software.
00015
00016 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00017 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00018 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00019 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00020 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00021 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00022 SOFTWARE.
00023 */
00024
00032 #include "threading/CtWorker.hpp"
00033
00034 CtWorker::CtWorker() : m_running(CT_FALSE) {
00035 }
00036
00037 CtWorker::~CtWorker() {
00038     joinTask();
00039 }
00040
00041 CtBool CtWorker::isRunning() {
00042     return m_running.load();
00043 }
00044
00045 void CtWorker::setRunning(CtBool running) {
00046     return m_running.store(running);
00047 }
00048
00049 void CtWorker::setTask(const CtTask& task, std::function<void()> callback) {
00050     alreadyRunningCheck();
00051     m_task = task;
00052     m_callback = callback;
00053 }
00054
00055 void CtWorker::runTask() {
00056     alreadyRunningCheck();
00057     setRunning(CT_TRUE);
00058     m_thread = std::thread([this]{
00059         m_task.getTaskFunc()();
00060         m_task.getCallbackFunc()();
00061         m_callback();
00062         setRunning(CT_FALSE);
00063     });
00064 }
00065
00066 void CtWorker::joinTask() {
00067     if (m_thread.joinable()) {
00068         m_thread.join();
00069     }
00070 }
00071
00072 void CtWorker::alreadyRunningCheck() {
00073     if (isRunning()) {
00074         throw CtWorkerError("CtWorker already running.");
00075     }
00076     joinTask();
00077 }
```

9.75 src/threading/CtWorkerPool.cpp File Reference

```
#include "threading/CtWorkerPool.hpp"
```

Include dependency graph for CtWorkerPool.cpp:



9.75.1 Detailed Description

Date

18-01-2024

Definition in file CtWorkerPool.cpp.

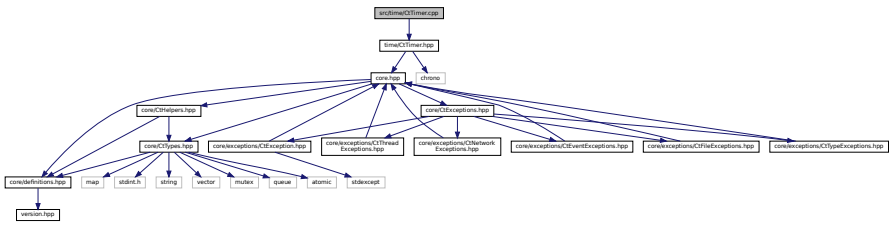
9.76 CtWorkerPool.cpp

```
00001 /*
00002 MIT License
00003
00004 Copyright (c) 2024 Mouzenidis Panagiotis
00005
00006 Permission is hereby granted, free of charge, to any person obtaining a copy
00007 of this software and associated documentation files (the "Software"), to deal
00008 in the Software without restriction, including without limitation the rights
00009 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00010 copies of the Software, and to permit persons to whom the Software is
00011 furnished to do so, subject to the following conditions:
00012
00013 The above copyright notice and this permission notice shall be included in all
00014 copies or substantial portions of the Software.
00015
00016 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00017 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00018 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00019 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00020 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00021 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00022 SOFTWARE.
00023 */
00024
00032 #include "threading/CtWorkerPool.hpp"
00033
00034 CtWorkerPool::CtWorkerPool(CtUInt32 nworkers) : m_nworkers(nworkers), m_active_tasks(0),
       m_queued_tasks(0) {
00035     for (int idx = 0; idx < m_nworkers; idx++) {
00036         m_workers.push_back(std::make_unique<CtWorker>());
00037     }
00038 }
00039
00040 CtWorkerPool::~CtWorkerPool() {
00041     CtThread::stop();
00042     free();
00043 }
00044
00045 void CtWorkerPool::addTask(const CtTask& task) {
00046     std::scoped_lock lock(m_mtx_control);
00047     m_tasks.push(task);
00048     m_queued_tasks++;
00049     try {
00050         start();
00051     } catch (const CtThreadError& e) {
00052     }
```

```
00053 }
00054
00055 void CtWorkerPool::join() {
00056     CtThread::join();
00057 }
00058
00059 void CtWorkerPool::assignTask(CtUInt32 idx) {
00060     std::scoped_lock lock(m_mtx_control);
00061     try {
00062         m_workers.at(idx).get()->setTask(m_tasks.front(), [this](){m_active_tasks--;});
00063         m_workers.at(idx).get()->runTask();
00064     } catch (const CtWorkerError& e) {
00065         return;
00066     }
00067     m_active_tasks++;
00068     m_tasks.pop();
00069     m_queued_tasks--;
00070 }
00071
00072 void CtWorkerPool::free() {
00073     for (int idx = 0; idx < m_nworkers; idx++) {
00074         m_workers.back().get()->joinTask();
00075         m_workers.pop_back();
00076     }
00077 }
00078
00079 void CtWorkerPool::loop() {
00080     CtBool s_breakFlag = CT_FALSE;
00081     while (!s_breakFlag) {
00082         for (int idx = 0; idx < m_nworkers; idx++) {
00083             if (!m_workers.at(idx).get()->isRunning() && m_queued_tasks.load() != 0) {
00084                 assignTask(idx);
00085             }
00086         }
00087         s_breakFlag = (m_queued_tasks.load() == 0) && (m_active_tasks.load() == 0);
00088     }
00089     setRunning(CT_FALSE);
00090 }
```

9.77 src/time/CtTimer.cpp File Reference

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



9.77.1 Detailed Description

Date

18-01-2024

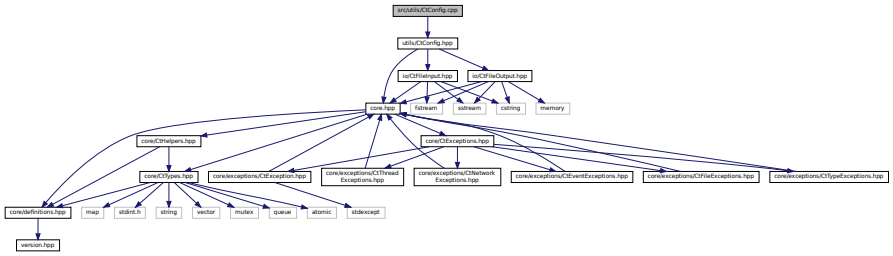
Definition in file CtTimer.cpp.

9.78 CtTimer.cpp

```
00001 /*
00002 MIT License
00003
00004 Copyright (c) 2024 Mouzenidis Panagiotis
00005
00006 Permission is hereby granted, free of charge, to any person obtaining a copy
00007 of this software and associated documentation files (the "Software"), to deal
00008 in the Software without restriction, including without limitation the rights
00009 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00010 copies of the Software, and to permit persons to whom the Software is
00011 furnished to do so, subject to the following conditions:
00012
00013 The above copyright notice and this permission notice shall be included in all
00014 copies or substantial portions of the Software.
00015
00016 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00017 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00018 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00019 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00020 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00021 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00022 SOFTWARE.
00023 */
00024
00032 #include "time/CtTimer.hpp"
00033
00034 CtTimer::CtTimer() {
00035     m_reference = 0;
00036 }
00037
00038 CtTimer::~CtTimer() {
00039 }
00040
00041 void CtTimer::tic() {
00042     m_reference = current();
00043 }
00044
00045 CtUInt64 CtTimer::toc() {
00046     return current() - m_reference;
00047 }
00048
00049 CtUInt64 CtTimer::current() {
00050     return std::chrono::duration_cast<std::chrono::milliseconds>(
00051         std::chrono::high_resolution_clock::now().time_since_epoch()
00052     ).count();
00053 }
```

9.79 src/utils/CtConfig.cpp File Reference

`#include "utils/CtConfig.hpp"`
Include dependency graph for CtConfig.cpp:



9.79.1 Detailed Description

Date

10-03-2024

Definition in file CtConfig.cpp.

9.80 CtConfig.cpp

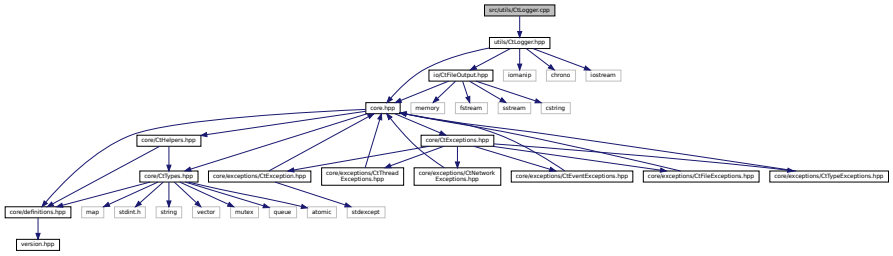
```
00001 /*
00002 MIT License
00003
00004 Copyright (c) 2024 Mouzenidis Panagiotis
00005
00006 Permission is hereby granted, free of charge, to any person obtaining a copy
00007 of this software and associated documentation files (the "Software"), to deal
00008 in the Software without restriction, including without limitation the rights
00009 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00010 copies of the Software, and to permit persons to whom the Software is
00011 furnished to do so, subject to the following conditions:
00012
00013 The above copyright notice and this permission notice shall be included in all
00014 copies or substantial portions of the Software.
00015
00016 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00017 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00018 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00019 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00020 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00021 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00022 SOFTWARE.
00023 */
00024
00032 #include "utils/CtConfig.hpp"
00033
00034 CtConfig::CtConfig(const CtString& configFile) : m_configFile(configFile) {
00035     m_source = nullptr;
00036     m_sink = nullptr;
00037 }
00038
00039 CtConfig::~CtConfig() {
00040     if (m_source != nullptr) {
00041         delete m_source;
00042     }
00043     if (m_sink != nullptr) {
00044         delete m_sink;
00045     }
00046 }
00047
00048 void CtConfig::read() {
00049     std::scoped_lock lock(m_mtx_control);
00050     m_source = new CtFileInput(m_configFile);
00051     m_source->setDelimiter("\n", 1);
00052
00053     CtRawData data(512);
00054
00055     while(m_source->read(&data)) {
00056         parseLine(CtString((CtChar*)data.get(), data.size()));
00057         data.reset();
00058     }
00059
00060     delete m_source;
00061     m_source = nullptr;
00062 }
00063
00064 void CtConfig::write() {
00065     std::scoped_lock lock(m_mtx_control);
00066     m_sink = new CtFileOutput(m_configFile, CtFileOutput::WriteMode::Truncate);
00067     m_sink->setDelimiter("\n", 1);
00068     CtMap<CtString, CtString>::iterator iter;
00069
00070     CtRawData data(512);
00071     for (iter = m_configValues.begin(); iter != m_configValues.end(); ++iter) {
00072         CtString line = iter->first + CtString(" = ") + iter->second;
00073         data.clone((CtUInt8*)line.c_str(), line.size());
00074         m_sink->write(&data);
00075         data.reset();
00076     }
00077
00078     delete m_sink;
00079     m_sink = nullptr;
00080 }
00081
00082 void CtConfig::parseLine(const CtString& line) {
00083     size_t separatorPos = line.find('=');
00084     size_t commentPos = line.find('#');
00085     size_t eol = line.size();
00086     CtBool hasComment = commentPos != CtString::npos;
00087     CtBool hasSeparator = separatorPos != CtString::npos;
00088
00089     if (hasSeparator && hasComment) {
00090         if (separatorPos > commentPos) {
00091             throw CtFileParseError("Invalid comment.");
00092         } else {
```

```
00093             eol = commentPos;
00094         }
00095     } else if (!hasSeparator && !hasComment) {
00096         throw CtFileParseError("Invalid line entry.");
00097     } else if (!hasSeparator && hasComment) {
00098         return;
00099     }
00100
00101     CtString key = line.substr(0, separatorPos);
00102     CtString value = line.substr(separatorPos + 1, eol - (separatorPos + 1));
00103
00104     key.erase(0, key.find_first_not_of(" \t\r\n"));
00105     key.erase(key.find_last_not_of(" \t\r\n") + 1);
00106     value.erase(0, value.find_first_not_of(" \t\r\n"));
00107     value.erase(value.find_last_not_of(" \t\r\n") + 1);
00108
00109     m_configValues[key] = value;
00110 }
00111
00112 CtInt32 CtConfig::parseAsInt(const CtString& key) {
00113     CtString str_value = getValue(key);
00114     return CtStringHelpers::StrToInt(str_value);
00115 }
00116
00117 CtUInt32 CtConfig::parseAsUInt(const CtString& key) {
00118     CtString str_value = getValue(key);
00119     return CtStringHelpers::StrToUInt(str_value);
00120 }
00121
00122 CtFloat CtConfig::parseAsFloat(const CtString& key) {
00123     CtString str_value = getValue(key);
00124     return CtStringHelpers::StrToFloat(str_value);
00125 }
00126
00127 CtDouble CtConfig::parseAsDouble(const CtString& key) {
00128     CtString str_value = getValue(key);
00129     return CtStringHelpers::StrToDouble(str_value);
00130 }
00131
00132 CtString CtConfig::parseAsString(const CtString& key) {
00133     return getValue(key);
00134 }
00135
00136 void CtConfig::reset() {
00137     m_configValues.clear();
00138 }
00139
00140 CtString CtConfig::getValue(const CtString& key) {
00141     if (m_configValues.find(key) != m_configValues.end()) {
00142         return m_configValues[key];
00143     } else {
00144         throw CtKeyNotFoundError(CtString("Key <") + key + CtString("> not found."));
00145     }
00146 }
00147
00148 void CtConfig::writeInt(const CtString& p_key, const CtInt32& p_value) {
00149     writeString(p_key, ToCtString(p_value));
00150 }
00151
00152 void CtConfig::writeUInt(const CtString& p_key, const CtUInt32& p_value) {
00153     writeString(p_key, ToCtString(p_value));
00154 }
00155
00156 void CtConfig::writeFloat(const CtString& p_key, const CtFloat& p_value) {
00157     writeString(p_key, ToCtString(p_value));
00158 }
00159
00160 void CtConfig::writeDouble(const CtString& p_key, const CtDouble& p_value) {
00161     writeString(p_key, ToCtString(p_value));
00162 }
00163
00164 void CtConfig::writeString(const CtString& p_key, const CtString& p_value) {
00165     m_configValues[p_key] = p_value;
00166 }
```

9.81 src/utils/CtLogger.cpp File Reference

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

Include dependency graph for CtLogger.cpp:



9.81.1 Detailed Description

Date

10-03-2024

Definition in file CtLogger.cpp.

9.82 CtLogger.cpp

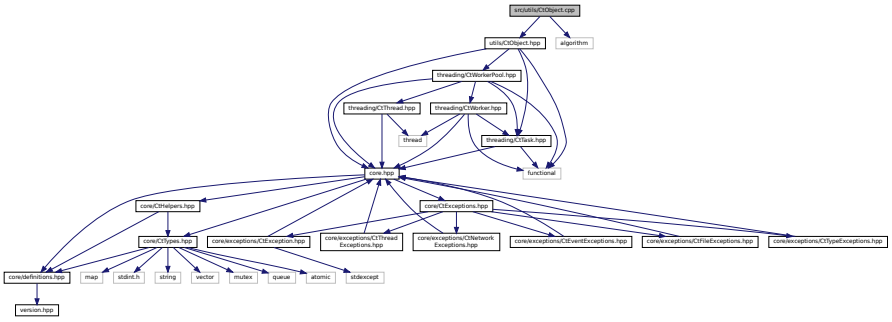
```
00001 /*
00002 MIT License
00003
00004 Copyright (c) 2024 Mouzenidis Panagiotis
00005
00006 Permission is hereby granted, free of charge, to any person obtaining a copy
00007 of this software and associated documentation files (the "Software"), to deal
00008 in the Software without restriction, including without limitation the rights
00009 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00010 copies of the Software, and to permit persons to whom the Software is
00011 furnished to do so, subject to the following conditions:
00012
00013 The above copyright notice and this permission notice shall be included in all
00014 copies or substantial portions of the Software.
00015
00016 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00017 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00018 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00019 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00020 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00021 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00022 SOFTWARE.
00023 */
00024
00032 #include "utils/CtLogger.hpp"
00033
00034 CtLogger::CtLogger(CtLogger::Level level, const CtString& componentName) : m_level(level),
       m_componentName(componentName) {
00035 }
00036
00037 CtLogger::~CtLogger() {
00038 }
00039
00040 void CtLogger::log_debug(const CtString& message) {
00041     log(CtLogger::Level::DEBUG, message);
00042 }
00043
00044 void CtLogger::log_info(const CtString& message) {
00045     log(CtLogger::Level::INFO, message);
00046 }
00047
00048 void CtLogger::log_warning(const CtString& message) {
00049     log(CtLogger::Level::WARNING, message);
00050 }
00051
00052 void CtLogger::log_error(const CtString& message) {
00053     log(CtLogger::Level::ERROR, message);
```

```
00054 }
00055
00056 void CtLogger::log_critical(const CtString& message) {
00057     log(CtLogger::Level::CRITICAL, message);
00058 }
00059
00060 void CtLogger::log(CtLogger::Level level, const CtString& message) {
00061     std::scoped_lock lock(m_mtx_control);
00062     if (level >= m_level) {
00063         CtString logEntry = generateLoggerMsg(level, m_componentName, message);
00064         std::cout « logEntry « std::endl;
00065     }
00066 }
00067
00068 const CtString CtLogger::generateLoggerMsg(CtLogger::Level level, const CtString& componentName, const
      CtString& message) {
00069     std::chrono::system_clock::time_point now = std::chrono::system_clock::now();
00070     std::time_t now_c = std::chrono::system_clock::to_time_t(now);
00071     std::stringstream timestamp;
00072     timestamp « std::put_time(std::localtime(&now_c), "%Y-%m-%d %X");
00073
00074     return CtString("[" + timestamp.str() + "] [" + levelToString(level) + "] " + componentName + ": "
      + message);
00075 }
00076
00077 const CtString CtLogger::levelToString(CtLogger::Level level) {
00078     CtString levelStr;
00079     switch (level) {
00080         case CtLogger::Level::DEBUG:
00081             levelStr = "DEBUG";
00082             break;
00083         case CtLogger::Level::INFO:
00084             levelStr = "INFO";
00085             break;
00086         case CtLogger::Level::WARNING:
00087             levelStr = "WARNING";
00088             break;
00089         case CtLogger::Level::ERROR:
00090             levelStr = "ERROR";
00091             break;
00092         case CtLogger::Level::CRITICAL:
00093             levelStr = "CRITICAL";
00094             break;
00095         default:
00096             levelStr = "DEBUG";
00097             break;
00098     }
00099
00100     return levelStr;
00101 }
```

9.83 src/utils/CtObject.cpp File Reference

```
#include "utils/CtObject.hpp"
#include <algorithm>
```

Include dependency graph for CtObject.cpp:



9.83.1 Detailed Description

Date

02-02-2024

Definition in file CtObject.cpp.

9.84 CtObject.cpp

```
00001 /*
00002 MIT License
00003
00004 Copyright (c) 2024 Mouzenidis Panagiotis
00005
00006 Permission is hereby granted, free of charge, to any person obtaining a copy
00007 of this software and associated documentation files (the "Software"), to deal
00008 in the Software without restriction, including without limitation the rights
00009 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00010 copies of the Software, and to permit persons to whom the Software is
00011 furnished to do so, subject to the following conditions:
00012
00013 The above copyright notice and this permission notice shall be included in all
00014 copies or substantial portions of the Software.
00015
00016 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00017 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00018 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00019 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00020 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00021 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00022 SOFTWARE.
00023 */
00024
00032 #include "utils/CtObject.hpp"
00033
00034 #include <algorithm>
00035
00036 CtObject::CtObject() : m_pool(1) {
00037 }
00038
00039 CtObject::~CtObject() {
00040     m_pool.join();
00041 }
00042
00043 void CtObject::connectEvent(CtObject* p_obj, CtUInt32 p_eventCode, CtTask& p_task) {
00044     p_obj->connectEvent(p_eventCode, p_task);
00045 }
00046
00047 void CtObject::waitPendingEvents() {
00048     m_pool.join();
00049 }
00050
00051 void CtObject::connectEvent(CtUInt32 p_eventCode, CtTask& p_task) {
00052     std::scoped_lock lock(m_mtx_control);
00053     if (!hasEvent(p_eventCode)) {
00054         throw CtEventNotExistsError("Event is not registed. " + ToCtString(p_eventCode));
00055     }
00056     m_triggers.insert({p_eventCode, p_task});
00057 }
00058
00059 void CtObject::triggerEvent(CtUInt32 p_eventCode) {
00060     std::scoped_lock lock(m_mtx_control);
00061     if (!hasEvent(p_eventCode)) {
00062         throw CtEventNotExistsError("Event is not registed. " + ToCtString(p_eventCode));
00063     }
00064     std::pair<CtMultiMap<CtUInt32, CtTask>::iterator, CtMultiMap<CtUInt32, CtTask>::iterator>
      s_iterRange;
00065     s_iterRange = m_triggers.equal_range(p_eventCode);
00066
00067     CtMultiMap<CtUInt32, CtTask>::iterator s_iter;
00068     for (s_iter = s_iterRange.first; s_iter != s_iterRange.second; ++s_iter) {
00069         m_pool.addTask(s_iter->second);
00070     }
00071 }
00072
00073 void CtObject::registerEvent(CtUInt32 p_eventCode) {
00074     std::scoped_lock lock(m_mtx_control);
00075     if (hasEvent(p_eventCode)) {
00076         throw CtEventAlreadyExistsError("Event is already registed.");
00077     }
00078     m_events.push_back(p_eventCode);
00079 }
```

```
00080
00081 CtBool CtObject::hasEvent(CtUInt32 p_eventCode) {
00082     return std::any_of(m_events.begin(), m_events.end(), [&p_eventCode](CtUInt8 s_event) {
00083         return (s_event == p_eventCode);
00084     });
00085 }
```

Index