

# CrossSockets protocol



Project: CrossSockets protocol  
Author: Jeroen Saey  
Startdate: 26-01-2013  
Version: 1.0

## Copyright

©SuperSmash.nl, SuperSmash  
The CrossSocket protocol is copyrighted by SuperSmash.  
This framework may not be used without acknowledgement of the creator.  
You are not allowed to modify the SuperSmash Framework in any way.













The creator is not responsible for any damage this protocol can cause while using irresponsible.  
This protocol may not be modified for personal use and may not be sold as your own.

## Revision history

Date	Version	Description	Author
26-01-2013	1.0	C++ CrossSockets protocol launched	Jeroen Saey

Color information throughout the document.

Color	Icon	Description
blue		Used for namespaces
orange		Used for classes
red		Used for critical information
purple		Used for warning information
brown		Used for constants
green		Used for public functions
green		Used for private functions
green		Used for protected functions
green		Used for virtual functions
green		Used for enums



# Caution:

Making changes to the C++ CrossSocket protocol can prevent it from working:

It is not allowed to change or modify the crossSocket protocol in any way. It is NOT OK to take it, copyright under your name (or your group's name), and sell it as your own.

It is NOT OK to take it, revise it, copyright it and then sell it as your own.

## Table of Contents

1. Preface.....	5
1.1 Scope .....	5
1.2 Audience.....	5
1.3 Purpose.....	5
2. Introduction.....	6
3. System design.....	7
3.1 Communication flow between components.....	7
5. File information .....	8
5.1 CrossSockets.....	8
5.1.1 BaseCrossSocket.....	8
5.1.2 CrossSocket .....	10
5.1.3 CrossSocket config.....	10
5.1.4 CrossSocket Errors.....	11
5.2 CrossThreads .....	12
5.2.1 CrossThreads .....	12
5.2.2 CrossThreadsHandler .....	12



## 1. Preface

### 1.1 Scope

This document will be used as a reference to the CrossSockets protocol. This document will explain all the functions that are inside this protocol.

### 1.2 Audience

The target audiences for this document are the developers who need to create client/server applications. . (For Windows and Linux environments)

### 1.3 Purpose

This document describes the technical design and contains information necessary for technical understanding of the CrossSockets protocol.



## 2. Introduction

If you need to create a client/server connection between applications it can be hard to do so. The CrossSockets protocol was created in mind that it doesn't need to be that hard.

The CrossSockets protocol uses the Winsock2 and the POSIX socket libraries to create the connections. This protocol also uses threading functionalities so multiple clients can connect to the server.

The CrossSockets protocol is a set of functions you can use as a programmer. These functions will help you to create client/server applications.

**Because coding needs to be simple!**



## 3. System design

### 3.1 Communication flow between components



When a client connects to the server, the client can be asked to enter a password (if configured). This password needs to be entered correctly before the client-server connection is established successfully.

The requests are sent to the server and are processed by the socketConnection libraries that the CrossSocket protocol uses .

# 5. File information



## 5.1 CrossSockets



### 5.1.1 BaseCrossSocket

This file creates several virtual functions that can be accessed throughout the CrossSockets protocol. (If the specified class extends the BaseSocket class)


The `BaseCrossSocket` function  will be used as the constructor.



The `~BaseCrossSocket` function   will be used as the destructor.



The `blockTypeEnum` enum  will be used a collection that consist of several blocking types for the BaseSocket class .



The `ioTypeEnum` enum  be used a collection that consist of several input/output streamtypes for the BaseSocket class .

The `listen` function   will listen for socket connections.



The `accept` function   will accept a new socketConnection (client).



The `isConnected` function   will check if the current socket is connected and if so, return true.



The `checkPassword` function   will check if the password for the server is correct (if specified).



The `closeSocketHandle` function   will forcibly close the client socket and returns a int.



The `write` function   (consisting of a message) will write to the socket and returns a int.



The `write` function   (consisting of a message and length) will write to the socket (by a specified length) and returns a int.



The `writeLine` function   (consisting of a message) will write a line to the socket and returns a int.

The `writeLine` function   (consisting of a message and length) will write a line to the socket (by a specified length) and returns a int.

The `read` function   (consisting of a buffer) will read all bytes into the buffer and returns a `ssize_t`

The `read` function   (consisting of a buffer and bytes) will read some bytes (specified by the bytes) into the buffer and returns a `ssize_t`.

The `readLine` function   (consisting of a buffer and bytes) will read a line (specified by the bytes) into the buffer and returns a `ssize_t`.



The `readLine` function   (consisting of a size) will read a line (specified by the size) into the buffer and returns the string that has been read from the buffer's content.







# CrossSockets protocol { 9 of 12 } © SuperSmash



C++



<http://www.SuperSmash.nl>

The `setPassword` function   will set the password that needs to be used by the server to authenticate clients.

The `getClientSocket` function   will return a int that is used to identify the clients sockethandle.

The `getServerHost` function   will try to get the serverHost information and return true if the information can be requested. (else if returns an error)

The `getClientHost` function   will try to get the clientHost information and return true if the information can be requested. (else if returns an error)

The `setTimeout` function   will set the waiting timeout for the server. (can only be used in *blockingmode*)



The `printError` function   will print the error to the console.

The `getError` function   will get the current error.



The `getSocket` function   will get the current socket.

The `create` function   will create a socket based on the given socket descriptor. (int)



The `reset` function   will reset the current socket information.

The `waitIO` function   will wait a while before selecting the correct IO.

The `waitRead` function   will wait a while before reading from the IO.

The `waitWrite` function   will wait a while before writing to the IO.



The `handleError` function   will handle the current error(s).

The `noError` function   will set the current error(s) to none.


The `setError` function   will set the current error to a given error (with error-message).





## 5.1.2 CrossSocket


The CrossSocket file will hold the functions to actually create a CrossSocket. This CrossSocket can be a client or a server socket. The CrossSocket class  extends the base: BaseCrossSocket class 


This file creates several functions.


The `CrossSocket`  function will be used as a constructor.


The `~CrossSocket`  function will be used as a destructor.


The `bind` function  (consisting of a port) will bind the socket and binds the socket on a specified port. (You can use 0 to bind to any of the available ports)


The `bind` function  (consisting of a port and a hostName) will bind the socket and binds the socket on a specified port and hostName. (You can also use the host as an interface identifier)


The `connect` function  will connect the socket to a specified hostname and port.


The `getClientAddress` function  will return the current address of the client.

The `getClientPort` function  will return the current port of the client.

The `getClientHostName` function  will return the current hostname of the client.

The `getServerAddress` function  will return the current address of the server.

The `getServerPort` function  will return the current port of the server.

The `getServerHostName` function  will return the current hostname of the server.

The `getSocket` function  will retrieve the current Socket-information.

The `create` function  will create a socket based on the given socket descriptor. (int)

## 5.1.3 CrossSocket config


The CrossSocket config file will hold the configurations for the CrossSocket protocol.



### Constants:

Name:	Value:	Description:
ERRORMODE	0	Set the errorMode to a specified mode 0: verbose error 1: throw error




## 5.1.4 CrossSocket Errors

This file will hold the functions and enums for the error that can occur in the CrossSocket class .



The `CrossSocketErrorsEnum` enum  will hold all the error Types for the CrossSocket class .



```
ok,           // Operation successful
fatal,        // Unspecified error
notReady,     // The socket was not ready (blockMode)
portInUse,    // The specified port is already in use
notConnected, // The socket is invalid or not connected
messageTooLong // The messageSize is too big to be send
terminated,   // Connection terminated (by client)
noResponse,   // Cannot connect to client
timeout,      // Read/Write operation timeout occurred (in blockingMode)
interrupted   // Operation was blocked by signal
```

The `CrossSocketErrors` function  will be used as a constructor.

The `CrossSocketErrors` function  (consisting of a `errorEnum` type) will be used as a constructor.

The `~CrossSocketErrors` function   will be used as a deconstructor.

The `getError` function   will be used to get the current Error.

The `getFailedClass` function   will be used to get the class where the error occurred.

The `setErrorString` function   will be used to set an error message.

The `setFailedClass` function   will be used to set the failed class.


## 5.2 CrossThreads


### 5.2.1 CrossThreads


The CrossThreads class makes it easier to create threads. (For Windows and Linux environments)  
This file creates several functions :


The `CrossThreads` function  will be used as a constructor.


The `~CrossThreads` function   will be used as a destructor.


The `setThreadCallback` function  will be used to set a function callback for the thread that has been created.


The `getThreadID` function  will be used to return the current thread ID (as a long)

The `create` function  will be used to create the thread.

The `remove` function  will be used to remove the thread.

The `join` function  will be used to join the thread.

The `isCreated` function  will be used to check if the thread has been created.

The `stop` function  will be used to stop the current thread;


### 5.2.2 CrossThreadsHandler


The CrossThreadsHandler class is used as a wrapper to create a thread and hold a threadList with all the threads created. (Use this class to create a thread do NOT use the CrossThreads class directly)  
This file creates several functions :

The `CrossThreadsHandler` function  will be used as a constructor.

The `~CrossThreadsHandler` function   will be used as a destructor.

The `createAndStartThread` function  will be used to create and start the thread

The `removeThread` function  will be used to remove a thread specified by the threadID.

The `joinThread` function  will be used to join a thread specified by the threadID.

