SFML

* [Main Page](http://docs.google.com/index.htm)
* [Modules](http://docs.google.com/modules.htm)
* [Classes](http://docs.google.com/annotated.htm)
* [Files](http://docs.google.com/files.htm)
* [Class List](http://docs.google.com/annotated.htm)
* [Class Index](http://docs.google.com/classes.htm)
* [Class Hierarchy](http://docs.google.com/hierarchy.htm)
* [Class Members](http://docs.google.com/functions.htm)
* **sf**
* [Socket](http://docs.google.com/classsf_1_1Socket.htm)

[Public Types](#_gjdgxs) | [Public Member Functions](#_30j0zll) | [Protected Types](#_1fob9te) | [Protected Member Functions](#_3znysh7) | [Friends](#_2et92p0) | [List of all members](http://docs.google.com/classsf_1_1Socket-members.htm)

sf::Socket Class Reference

[Network module](http://docs.google.com/group__network.htm)

Base class for all the socket types. [More...](http://docs.google.com/classsf_1_1Socket.htm#details)

#include <[Socket.hpp](http://docs.google.com/Socket_8hpp_source.htm)>

Inheritance diagram for sf::Socket:



| Public Types | |
| --- | --- |
| enum | [Status](http://docs.google.com/classsf_1_1Socket.htm#a51bf0fd51057b98a10fbb866246176dc) {  [Done](http://docs.google.com/classsf_1_1Socket.htm#a51bf0fd51057b98a10fbb866246176dca1de3a85bc56d3ae85b3d0f3cfd04ae90),  [NotReady](http://docs.google.com/classsf_1_1Socket.htm#a51bf0fd51057b98a10fbb866246176dca8554848daae98f996e131bdeed076c09),  [Disconnected](http://docs.google.com/classsf_1_1Socket.htm#a51bf0fd51057b98a10fbb866246176dcab215141f756acdc23c67fad149710eb1),  [Error](http://docs.google.com/classsf_1_1Socket.htm#a51bf0fd51057b98a10fbb866246176dca1dc9854433a28c22e192721179a2df5d)  } |
|  | Status codes that may be returned by socket functions. [More...](http://docs.google.com/classsf_1_1Socket.htm#a51bf0fd51057b98a10fbb866246176dc) |
|  | |
| enum | { [AnyPort](http://docs.google.com/classsf_1_1Socket.htm#a5deb2c955fd347259c3a20d27b2481aaa5a3c30fd128895403afc11076f461b19) = 0 } |
|  | Some special values used by sockets. [More...](http://docs.google.com/classsf_1_1Socket.htm#a5deb2c955fd347259c3a20d27b2481aa) |
|  | |

| Public Member Functions | |
| --- | --- |
| virtual | [~Socket](http://docs.google.com/classsf_1_1Socket.htm#a79a4b5918f0b34a2f8db449089694788) () |
|  | Destructor. |
|  | |
| void | [setBlocking](http://docs.google.com/classsf_1_1Socket.htm#a165fc1423e281ea2714c70303d3a9782) (bool blocking) |
|  | Set the blocking state of the socket. |
|  | |
| bool | [isBlocking](http://docs.google.com/classsf_1_1Socket.htm#a0ec0d831b015e32eb5935fd2a9f8c67c) () const |
|  | Tell whether the socket is in blocking or non-blocking mode. |
|  | |

| Protected Types | |
| --- | --- |
| enum | [Type](http://docs.google.com/classsf_1_1Socket.htm#a5d3ff44e56e68f02816bb0fabc34adf8) {  [Tcp](http://docs.google.com/classsf_1_1Socket.htm#a5d3ff44e56e68f02816bb0fabc34adf8acc02e97e90234b957eaad4dff7f22214),  [Udp](http://docs.google.com/classsf_1_1Socket.htm#a5d3ff44e56e68f02816bb0fabc34adf8a6ebf3094830db4820191a327f3cc6ce2)  } |
|  | Types of protocols that the socket can use. [More...](http://docs.google.com/classsf_1_1Socket.htm#a5d3ff44e56e68f02816bb0fabc34adf8) |
|  | |

| Protected Member Functions | |
| --- | --- |
|  | [Socket](http://docs.google.com/classsf_1_1Socket.htm#a80ffb47ec0bafc83af019055d3e6a303) ([Type](http://docs.google.com/classsf_1_1Socket.htm#a5d3ff44e56e68f02816bb0fabc34adf8) type) |
|  | Default constructor. |
|  | |
| SocketHandle | [getHandle](http://docs.google.com/classsf_1_1Socket.htm#ac0c63b13e61da8294bf54e888e97f9a3) () const |
|  | Return the internal handle of the socket. |
|  | |
| void | [create](http://docs.google.com/classsf_1_1Socket.htm#aafbe140f4b1921e0d19e88cf7a61dcbc) () |
|  | Create the internal representation of the socket. |
|  | |
| void | [create](http://docs.google.com/classsf_1_1Socket.htm#af1dd898f7aa3ead7ff7b2d1c20e97781) (SocketHandle handle) |
|  | Create the internal representation of the socket from a socket handle. |
|  | |
| void | [close](http://docs.google.com/classsf_1_1Socket.htm#a71f2f5c2aa99e01cafe824fee4c573be) () |
|  | Close the socket gracefully. |
|  | |

| Friends | |
| --- | --- |
| class | **SocketSelector** |
|  | |

## Detailed Description

Base class for all the socket types.

This class mainly defines internal stuff to be used by derived classes.

The only public features that it defines, and which is therefore common to all the socket classes, is the blocking state. All sockets can be set as blocking or non-blocking.

In blocking mode, socket functions will hang until the operation completes, which means that the entire program (well, in fact the current thread if you use multiple ones) will be stuck waiting for your socket operation to complete.

In non-blocking mode, all the socket functions will return immediately. If the socket is not ready to complete the requested operation, the function simply returns the proper status code ([Socket::NotReady](http://docs.google.com/classsf_1_1Socket.htm#a51bf0fd51057b98a10fbb866246176dca8554848daae98f996e131bdeed076c09)).

The default mode, which is blocking, is the one that is generally used, in combination with threads or selectors. The non-blocking mode is rather used in real-time applications that run an endless loop that can poll the socket often enough, and cannot afford blocking this loop.

See Also[sf::TcpListener](http://docs.google.com/classsf_1_1TcpListener.htm), [sf::TcpSocket](http://docs.google.com/classsf_1_1TcpSocket.htm), [sf::UdpSocket](http://docs.google.com/classsf_1_1UdpSocket.htm)

Definition at line [45](http://docs.google.com/Socket_8hpp_source.htm#l00045) of file [Socket.hpp](http://docs.google.com/Socket_8hpp_source.htm).

## Member Enumeration Documentation

| anonymous enum |
| --- |

Some special values used by sockets.

**Enumerator:**

| *AnyPort* | Special value that tells the system to pick any available port. |
| --- | --- |

Definition at line [65](http://docs.google.com/Socket_8hpp_source.htm#l00065) of file [Socket.hpp](http://docs.google.com/Socket_8hpp_source.htm).

| enum [sf::Socket::Status](http://docs.google.com/classsf_1_1Socket.htm#a51bf0fd51057b98a10fbb866246176dc) |
| --- |

Status codes that may be returned by socket functions.

**Enumerator:**

| *Done* | The socket has sent / received the data. |
| --- | --- |
| *NotReady* | The socket is not ready to send / receive data yet. |
| *Disconnected* | The TCP socket has been disconnected. |
| *Error* | An unexpected error happened. |

Definition at line [53](http://docs.google.com/Socket_8hpp_source.htm#l00053) of file [Socket.hpp](http://docs.google.com/Socket_8hpp_source.htm).

| | enum [sf::Socket::Type](http://docs.google.com/classsf_1_1Socket.htm#a5d3ff44e56e68f02816bb0fabc34adf8) | | --- | | protected |
| --- | --- | --- |

Types of protocols that the socket can use.

**Enumerator:**

| *Tcp* | TCP protocol. |
| --- | --- |
| *Udp* | UDP protocol. |

Definition at line [113](http://docs.google.com/Socket_8hpp_source.htm#l00113) of file [Socket.hpp](http://docs.google.com/Socket_8hpp_source.htm).

## Constructor & Destructor Documentation

| | virtual sf::Socket::~Socket | ( |  | ) |  | | --- | --- | --- | --- | --- | | virtual |
| --- | --- | --- | --- | --- | --- | --- |

Destructor.

| | sf::Socket::Socket | ( | [Type](http://docs.google.com/classsf_1_1Socket.htm#a5d3ff44e56e68f02816bb0fabc34adf8) | *type* | ) |  | | --- | --- | --- | --- | --- | --- | | protected |
| --- | --- | --- | --- | --- | --- | --- | --- |

Default constructor.

This constructor can only be accessed by derived classes.

Parameters

| type | Type of the socket (TCP or UDP) |
| --- | --- |

## Member Function Documentation

| | void sf::Socket::close | ( |  | ) |  | | --- | --- | --- | --- | --- | | protected |
| --- | --- | --- | --- | --- | --- | --- |

Close the socket gracefully.

This function can only be accessed by derived classes.

| | void sf::Socket::create | ( |  | ) |  | | --- | --- | --- | --- | --- | | protected |
| --- | --- | --- | --- | --- | --- | --- |

Create the internal representation of the socket.

This function can only be accessed by derived classes.

| | void sf::Socket::create | ( | SocketHandle | *handle* | ) |  | | --- | --- | --- | --- | --- | --- | | protected |
| --- | --- | --- | --- | --- | --- | --- | --- |

Create the internal representation of the socket from a socket handle.

This function can only be accessed by derived classes.

Parameters

| handle | OS-specific handle of the socket to wrap |
| --- | --- |

| | SocketHandle sf::Socket::getHandle | ( |  | ) | const | | --- | --- | --- | --- | --- | | protected |
| --- | --- | --- | --- | --- | --- | --- |

Return the internal handle of the socket.

The returned handle may be invalid if the socket was not created yet (or already destroyed). This function can only be accessed by derived classes.

ReturnsThe internal (OS-specific) handle of the socket

| bool sf::Socket::isBlocking | ( |  | ) | const |
| --- | --- | --- | --- | --- |

Tell whether the socket is in blocking or non-blocking mode.

ReturnsTrue if the socket is blocking, false otherwise See Also[setBlocking](http://docs.google.com/classsf_1_1Socket.htm#a165fc1423e281ea2714c70303d3a9782)

| void sf::Socket::setBlocking | ( | bool | *blocking* | ) |  |
| --- | --- | --- | --- | --- | --- |

Set the blocking state of the socket.

In blocking mode, calls will not return until they have completed their task. For example, a call to Receive in blocking mode won't return until some data was actually received. In non-blocking mode, calls will always return immediately, using the return code to signal whether there was data available or not. By default, all sockets are blocking.

Parameters

| blocking | True to set the socket as blocking, false for non-blocking |
| --- | --- |

See Also[isBlocking](http://docs.google.com/classsf_1_1Socket.htm#a0ec0d831b015e32eb5935fd2a9f8c67c)

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

* [Socket.hpp](http://docs.google.com/Socket_8hpp_source.htm)

Copyright � Laurent Gomila  ::  Documentation generated by [doxygen](http://www.doxygen.org/)  ::